4/6

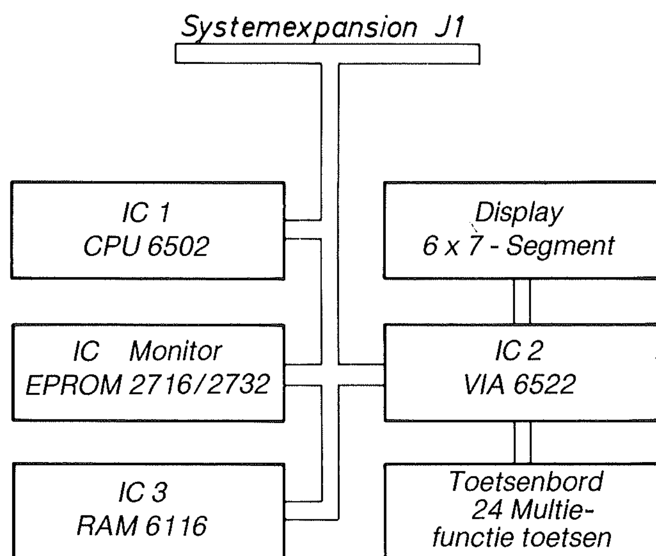
Microcomputer techniek

Inhoud

4/6.1

Microcomputer MPS 65

De MPS 65 is een microcomputer, die op een enkele Euro-formaatkaart is gebouwd (een zogenaamde single board computer).
Deze microcomputer kan op vele fronten worden ingezet, denk bijvoorbeeld aan het besturen van een modelbaan, maar ook voor het leren programmeren.
De MPS 65 is een relatief goedkope microcomputer, die zich uitstekend leent om een inzicht te krijgen in principe, opbouw en programmering van microcomputers. Het is een zelfstandig werkend systeem, dat op eenvoudige wijze is uit te breiden met andere systeemfuncties, door middel van op de uitbreidingsbus aan te sluiten Euro-kaarten.

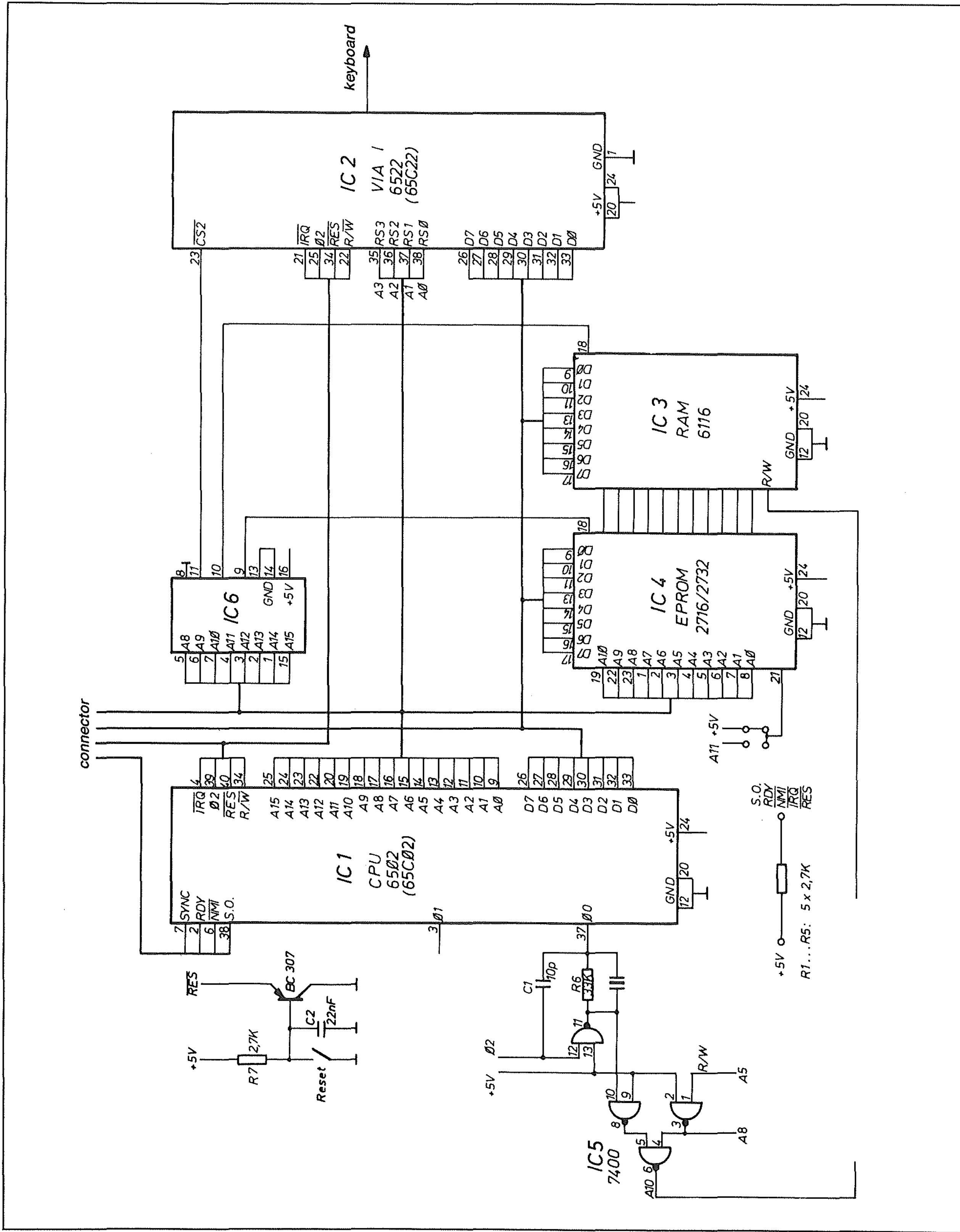


Figuur 4/6.1 -1:
MPS 65 Blokschema

De belangrijkste gegevens:

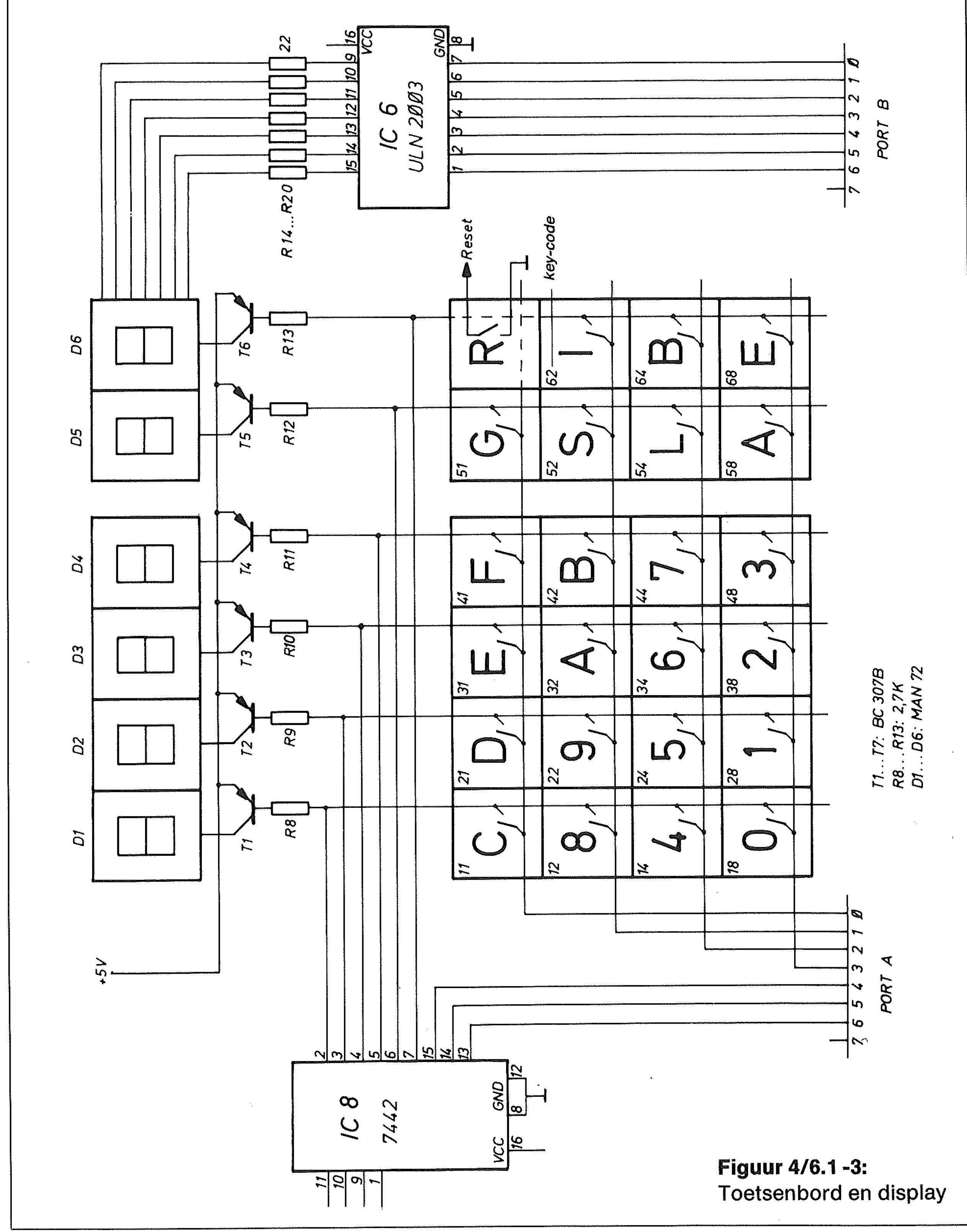
Microprocessor:	CPU 6502 (Rockwell)
Interface adaptor:	VIA 6522
Display:	6 stuks zevensegment displays
Werkgeheugen: (on board)	(2K Byte) RAM 6116
Programma-geheugen:	EPROM 2716/2732 (8 2k/4k) Monitor programma
Uitbreidingsbus:	SMP standaard uitbreidingsbus
Voeding:	sV/400 mA
Afmetingen:	10 cm x 16 cm kaart (Euro-formaat)

6.1 Microcomputer MPS 65



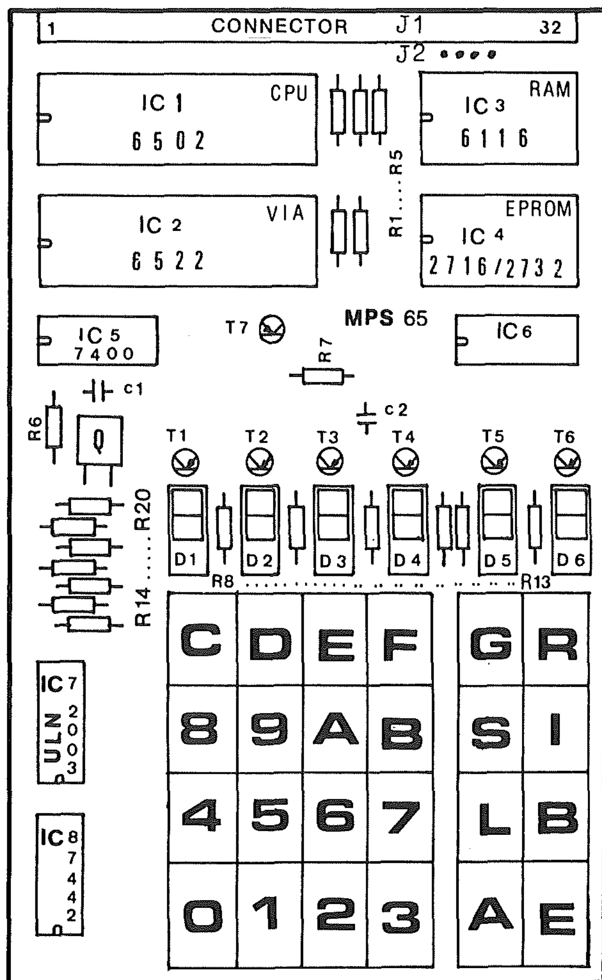
Figuur 4/6.1 -2: Schema

6.1 Microcomputer MPS 65

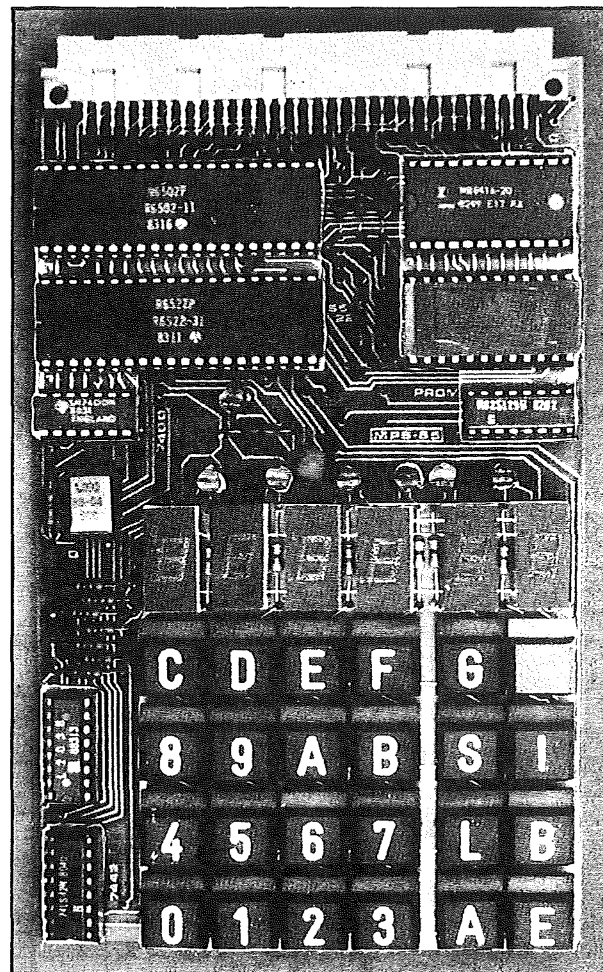


Figuur 4/6.1 -3:
Toetsenbord en display

6.1 Microcomputer MPS 65



Figuur 4/6.1 -4:
Onderdelen plattegrond

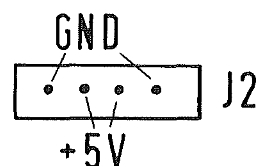


Figuur 4/6.1 -5:
De kant en klare Euro-kaart

Inschakelen van de MPS 65

Voedingsspanningsaansluiting

De MPS 65 microcomputer kan als zelfstandig systeem gebruikt worden. Daartoe moet een externe voeding van +5V gelijkspanning worden aangesloten op connector J2.



De spanning moet niet meer dan + of – 0,25V afwijken van 5V en de voeding moet in staat zijn continu 400 mA te leveren.

6.1 Microcomputer MPS 65

Wat te verwachten valt na het inschakelen!

Zoals van een geciviliseerd apparaat te verwachten is, stelt hij zich voor. Na het inschakelen verschijnt op het display

μPS-65

Na het indrukken van de toets "E" (=enter) verschijnt op het display 0200xx, waarbij xx de inhoud is van de geheugenplaats 0200. Na het inschakelen zal deze inhoud altijd 00 zijn, aangezien het monitorprogramma als eerste actie een geheugenreset uitvoert op het geheugenbereik 0200-07FF.

Opmerking!

1. Alle referenties aan adressen en data in dit hoofdstuk worden in de hexadecimale notatie gedaan, tenzij anders vermeld.
2. Als na inschakelen het display donker blijft, kan door het indrukken van de "R"-toets (Reset) de MPS 65 worden gedwongen opnieuw te starten.

TOETSFUNCTIES (de "keys" van het "keyboard")

Ø-F **HEX-toetsen**

Hexadecimale toetsen dienen voor de invoer van adressen en data in hexadecimale vorm (hexadecimaal = grondtal 17).
Voor representatie van de hexadecimale eenheden 10–15 wordt gebruik gemaakt van letters, waarbij A = 10, B = 11, C = 12, D = 13, E = 14, F = 15.

Na indrukken van de INDEX-toets krijgen de hexadecimale (kortweg hex) toetsen nog een andere functie. Deze is beschreven onder de beschrijving van de INDEX-toets.

E **ENTER-toets**

Deze toets heeft verschillende functies, te weten

1. Een stap voorwaarts (single step). Het indrukken heeft tot gevolg, dat het adres met 1 wordt verhoogd (geïncrementeerd). Uiteraard geeft het dataveld dan de inhoud van de nieuwe geheugenplaats weer.
2. Invoer van ingetoetste data in de aangegeven geheugenplaats, waarna de geheugenplaats met 1 wordt verhoogd.
3. Het beëindigen van de monitorprogramma's
   - Klok op tijd zetten (settime).
   - Tijd aangeven (displaytime).
   - BREAK-functie.

B **BACKSTEP-toets**

De BACKSTEP-toets heeft de functie, het adres met één te verminderen (decrement), zonder daarbij de in de geheugenplaats aanwezige data te wijzigen.

G **GO-toets**

Met de GO-toets wordt een gebruikersprogramma gestart, vanaf het op het display aangegeven adres.
Voor het veranderen cq. ingeven van het startadres van een gebruikersprogramma zie A-toets.

6.1 Microcomputer MPS 65

Na het starten van een gebruikersprogramma verschijnt op het display:

– – – – – – –

Vooropgesteld, dat het gebruikersprogramma zelf het display niet gebruikt.

Opmerking:
Het monitorprogramma en de systeemklok werken tijdens de uitvoering van een gebruikersprogramma op interruptbasis. Dat betekent, dat het toetsenbord actief blijft (via de routines KEYIN, KEYIN1) en de displays door het monitorprogramma worden gestuurd (via de adressen ADR16, ADR16+1, DAT8, resp. DISP1 t/m DISP6).

A **ADRES-toets**
Na indrukken van de A-toets worden de vier volgende HEX-toetsen geïnterpreteerd als een nieuw adres. Dit ingetoetste adres en de inhoud hiervan worden op het display zichtbaar, waarna alle andere functies weer beschikbaar zijn.

S **SAVE-toets**
Voor gebruik met uitbreidingskaart(en).

L **LOAD-toets**
Voor gebruik met uitbreidingskaart(en).

I **INDEX-toets**
Door middel van de INDEX-toets krijgt een aantal van de 16 HEX-toetsen een andere functie. Op het display verschijnt:

----In

De speciale functies zijn:

INDEX B	berekenen sprong offset vanaf absolute adressen.
INDEX C	verschuiven van datablok met één plaats om ruimte te maken voor een in te voegen instructie (werkt alleen in geheugenbereik Ø2ØØ-Ø7FF).
INDEX D	display-systeemtijd.
INDEX E	sprong naar, en programmastart vanaf, adres EØØØ.
INDEX F	klok op tijd zetten

BREAK-functie

Een interessante instructie van de 6502 microprocessor is de BREAK-instructie (code ØØ). Zodra de microprocessor een breakinstructie detecteert, wordt het lopende programma onderbroken en volgt een sprong (indirect, via adressen FFFE en FFFF) meestal naar een bepaalde routine in het monitorprogramma.
De BREAK-instructie is een software interrupt.
Bij het MPS 65 monitorprogramma, wordt na een BREAK de inhoud van de systeemregisters X, Y en A (Accumulator) weergegeven op het display.

Display: (X-Reg.) (Y-REG.) (ACCU)

De reactie op een BREAK kan echter door de gebruiker worden gewijzigd.

6.1 Microcomputer MPS 65

INDEXFUNCTIES

INDEX F

(Systeem)klok op tijd zetten.

Na indrukken van de INDEX-F-toets kan de systeemklok op tijd worden gezet, door met behulp van de toetsen 0-9 het uur en de minuten in te toetsen (in een 24 uurs klok). Na invoer van het vierde digit start de klok automatisch.
Door indrukken van de ENTER-toets verdwijnt het klokdisplay. De klok zelf echter loopt gewoon door. Alleen door de functies RESET, SAVE/LOAD wordt de klok tijdelijk onderbroken, na afloop van deze functies loopt hij echter weer normaal door.
In een gebruikersprogramma kan de tijd worden gebruikt door uitlezing van de adressen ØØA9, ØØAA en ØØAB, waarin respectievelijk de uren, minuten en seconden worden bewaard.

INDEX B

Berekenen van de sprongafstand (offset) tussen twee absolute adressen.

Branch-instructies (zoals BEQ, BPL enz.) zijn 2 byte-instructies, waarvan het eerste byte de instructiecode zelf is en het tweede byte de offset tussen het huidige adres en het doeladres.

Bijvoorbeeld:	code	mnemonic
	FØ	BEQ
	Ø9	9 stappen vooruit

Om deze offset op eenvoudige wijze te kunnen berekenen (vooral bij negatieve sprongen) kan de INDEX B-functie gebruikt worden. De instructie zelf bijv. BEQ = FØ wordt normaal ingetoetst en met ENTER ingevoerd.
Op de plaats waar de offset moet worden ingevoerd, wordt dan INDEX B gedrukt, waarna het absolute doeladres (in vier digits) wordt ingetoetst. Automatisch wordt dan de offset berekend, die nu met ENTER wordt ingevoerd.

Voorbeeld: Ø22Ø FØ ØE BEQ Ø23Ø
.
.
Ø23Ø Hier wordt in plaats van OE ingevoerd:
INDEX/B/Ø23Ø/ENTER

INDEX C

Data-blok verschuiven in geheugenbereik Ø2ØØ-Ø7FF.
Moet in een programma een instructie (of deel daarvan) worden ingevoegd, dan kan met behulp van INDEX C het gehele programma vanaf de op het display aangegeven geheugenplaats één plaats

6.1 Microcomputer MPS 65

worden opgeschoven. De open plaats die ontstaat wordt automatisch met een NOP = EA instructie gevuld, die echter daarna overgeschreven kan worden.
NB: Eventueel in het programma voorkomende absolute adresreferenties, die door de verschuiving niet meer kloppen worden *niet* automatisch aangepast.

INDEX D

Zet de lopende systeemtijd op het display, deze functie kan met de ENTER-toets worden beëindigd.

INDEX E

Het systeem voert een sprong uit naar adres EØØØ.
Een op adres E005 beginnend programma wordt door het systeem na initialisering gestart, als op de geheugenplaatsen EØØ3 en EØØ4 de codes Ø1 resp. FF staan.

Dus:	EØØ3	Ø1	
	EØØ4	FF	
	EØØ5		Startadres

De geheugenplaatsen EØØØ tot EØØ2 worden vrijgehouden, om daarvandaan met een JMP-instructie naar het begin van het programma te kunnen springen.
Toepassing: INDEX E.

FOUTMELDINGEN

Het MPS 65 systeem kan verschillende foutmeldingen geven, die met een nummer worden aangeduid.

Deze nummers hebben de volgende betekenis:

ERROR 1 Foutieve adresinvoer (beginadres > eindadres)
ERROR 2 Relatieve sprong te ver terug. Doeladres ligt meer dan 128 plaatsen terug.
ERROR 3 Relatieve sprong te ver naar voren. Doeladres ligt meer dan 127 plaatsen verder.
ERROR 4 Foutieve branch-instructie.
ERROR 6 Fout bij de afhandeling van interruptvectoren.

Na het indrukken van de ENTER-toets is een fout "gereset" en kan de correcte data ingetoetst worden.

Tabel 1: Monitorroutines van de MPS 65

Naam	Adres	Register	Beschrijving
KEYIN	F800	A	Uitlezen van toetsenbord met tijdsvertraging, de toetscode staat in de accu (contactdenden onderdrukt!)
KEYIN1	F803	A	Uitlezen van toetsenbord, zonder tijdsvertraging, de toetscode staat in de accu.
DISPEX	F806		Inschakelen van externe displaymode (voor besturing van de individuele elementen van de displays.
DISPSY	F809		Inschakelen van system displaymode (voor hexadecimale representatie van de "cijfers" 0-F.
PHAXY	F80C		Bewaren van de inhoud van Accu, X- en Y-register op de stack.
PLAXY	F80F	A,X,Y	Herstellen van de waarden van Accu, X- en Y-register met op de stack bewaarde (oude) inhouden.
ASK2	F812	A,X	Inlezen van een twee digits groot hexadecimaal getal. Het getal staat in adres 00A0.
ASK4	F815	A,X	Idem ASK2, maar voor 4 digits. Het getal staat in 00A0 en 00A1.
MON65	F818		Startadres van het MONITOR-programma van de MPS65, zonder initialisatie.
MON 65E	F81B		Startadres van het MONITOR-programma, waar naar toe wordt gesprongen na indrukken van de ENTER-toets.
T01S	F81E		Vertragingsroutine. Duur van de tijdsvertraging is 0,1 x de waarde in de accu in sec.
INC16X	F821		Een zestienbits getal wordt geïncrementeerd (met 1 verhoogd).
DEC16X	F824		Een zestienbits getal wordt gedecrementeerd (met 1 verlaagd).
ERR	F827	A,X,Y	Displaymelding ERROR (A). (A) is de waarde van de ACCU van 0-F.
STRICH	F82D		Display: ------
SUCHEX	F830	A	Uitlezing van het toetsenbord, met tijdsvertraging (de gedrukte toets staat in de Accu – 0-F).

Systeemroutines (beschrijving)

Een aantal van de monitorroutines kunnen door de gebruiker als subroutine worden aangeroepen in zijn programma's.

Veel gebruikte systeemfuncties kunnen door gebruikmaking van monitorroutines eenvoudig worden geïmplementeerd, waarbij men dan de zekerheid heeft dat daarin geen fouten zitten, terwijl deze routines in ieder geval met de systeemtiming rekening houden.

Voor de meeste routines fungeert de accumulator als parameteroverdrachtsregister (bijv. code van de ingedrukte toets bij routine KEYIN). X en Y register blijven, voorzover zij niet bij de parameteroverdracht zijn betrokken ongewijzigd (met uitzondering van ERR-F827).

Het monitorprogramma staat in PROM (programmable read only memory = niet vluchtig geheugen) vanaf adres F8ØØ. De routines worden door middel van een JSR-instructie aangeroepen.

De subroutineaanroepen worden via een sprongtabel (vanaf F8ØØ) naar het juiste startadres van de subroutines gedirigeerd (een soort indirecte subroutine aanroep).

Deze procedure heeft enkele voordelen, we noemen:

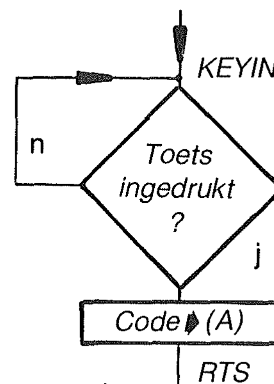
- de aanroepadressen liggen achter elkaar en zijn gemakkelijk te onthouden.
- de aanroepadressen veranderen niet bij aanpassing of verandering van het monitorprogramma.

Daardoor behoeven gebruikersprogramma's niet te worden aangepast bij wijziging van het monitorprogramma.

= = KEYIN = =

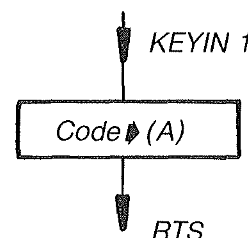
De aanroep van de routine KEYIN heeft tot gevolg, dat het systeem wacht tot er een toets ingedrukt wordt, dan wordt de code van deze toets in de accu gezet, waarna de terugkeer naar het hoofdprogramma volgt.

Verandering in register(s): geen



= = KEYIN 1 = =

De werking van deze subroutine is nagenoeg gelijk aan die van de vorige, zij het dan dat er niet wordt gewacht tot een toets ingedrukt is. (Voor toetscode zie tabel 2).

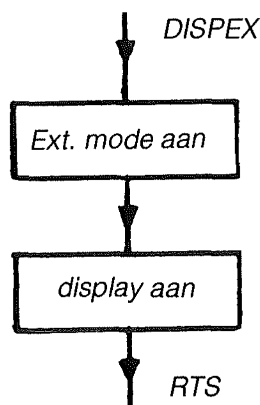


6.1 Microcomputer MPS 65

== DISPEX ==

Routine voor het omschakelen van de displaymode in externe aansturing. Hierbij staan de codes (het bit-patroon) voor elk van de 6 displays in geheugenlocaties DISP 1 . . . DISP6). (Voor bitpatronen en juiste adressen zie figuur 8)
Bij het starten van een programma door middel van de "GO"-toets wordt deze mode automatisch ingeschakeld.
DISPEX is dus alleen nodig als externe displaybesturing nodig is, nadat het gebruikersprogramma zelf via DISPSY de mode veranderd had.
DISPEX schakelt bovendien de displays weer aan, voor het geval zij uit waren.

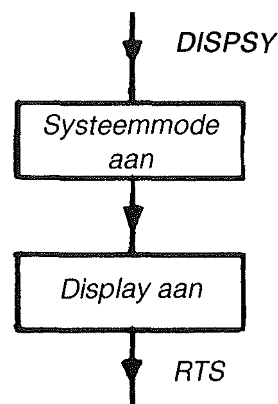
Verandering in register(s): geen



== DISPSY ==

Routine voor het omschakelen van de externe displaymode in systeembesturing.
Bij systeembesturing wordt de in de adressen ADR16, ADR16+1 en DAT8 staande data als hexadecimale waarde van respectievelijk telkens linker- en rechter-nibble (nibble = $\frac{1}{2}$ byte) dus 6 in totaal) in de displays 1 . . . 6 weergegeven. De subroutine verzorgt dus zelf de aansturing van de benodigde bits. DISPSY schakelt bovendien de displays aan indien zij uitgeschakeld waren.

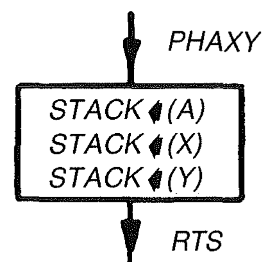
Verandering in register(s): geen



== PHAXY ==

Deze routine bewaart de inhouden van accumulator, het X-register en het Y-register op de stack.

Verandering in register(s): geen

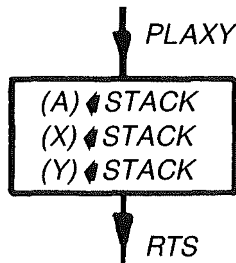


== PLAXY ==

Met deze routine kunnen de met PHAXY op de stack bewaarde registerinhouden weer in de betreffende registers worden teruggelezen.

Verandering in register(s): Accu, X en Y.

6.1 Microcomputer MPS 65



= = ASK2 = =

= = ASK4 = =

Routines voor het inlezen van een getal van resp. 2 of 4 digits.
Na afloop van de routine staan de ingelezen getallen in de geheugenplaatsen ADR16 en ADR16+1 (alleen gevuld door ASK4).

Verandering in register(s): A, X

N.B. De inhoud van geheugenplaatsen ADR16 en ADR16+1 worden door het systeem overschreven na een RESET, of na een JMP MON65 instructie.

= = MON65 = =

Dit is een adres voor een geprogrammeerde terugkeer naar het monitorprogramma, na afloop van een gebruikersprogramma.

De VIA wordt geïnitialiseerd en de stackpointer wordt gereset (op FF gezet).

Toepassing:

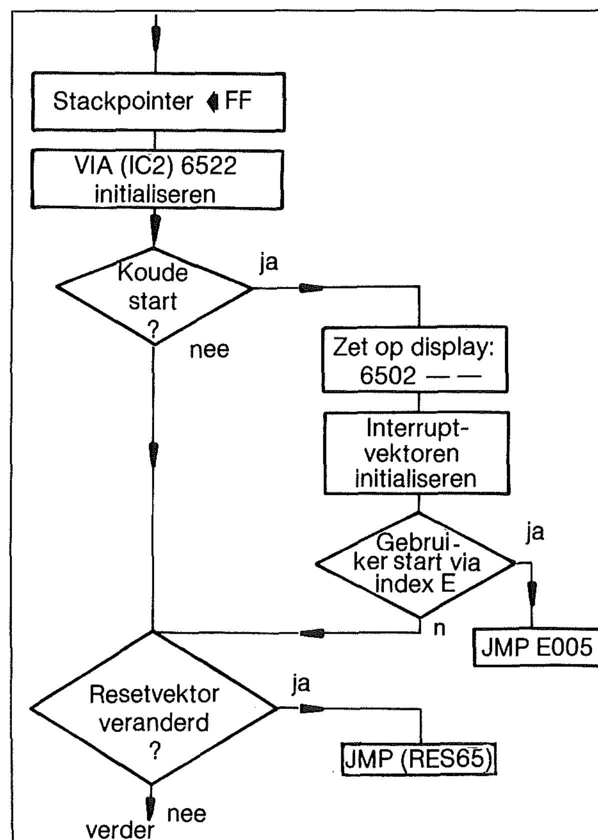
Beëindiging van een gebruikersprogramma door gebruik te maken van JMP MON65.

Gebruikersprogramma-instructies:

.

.

4C 18 F8 JMP MON65



Figuur 6:
Initialisering

6.1 Microcomputer MPS 65

= = MON 65E = =

Beëindigen van een gebruikersprogramma als boven, echter de laatste door het gebruikersprogramma op de displays gezette informatie mag niet worden overschreden.
Dit kan worden bereikt, door in plaats van een JMP MON65 een JMP MON65E uit te voeren als laatste instructie van het gebruikersprogramma.

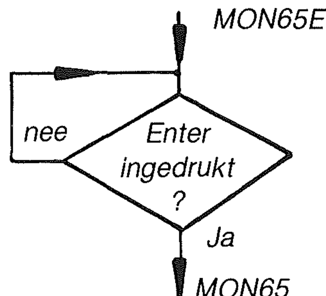
De monitor wacht nu op het indrukken van de ENTER-toets, alvorens de reinitialisatie uit te voeren en de displays weer te gebruiken voor adres en datadisplay van de monitor zelf.

Gebruikersprogramma-instructies:

.

.

4C 1B F8 JMP MON65E



= = TØ1S = =

Dit is een delay(tijdvertragings)-routine.
Aanroep van deze routine heeft tot gevolg, dat de terugkeer naar het hoofdprogramma (0,1 maal de inhoud van de accu) seconden duurt. Dat wil zeggen dat er tijdvertragingen van 0,1 tot 25,5 seconden (accu-inhoud hex Ø1 tot FF) mogelijk zijn.
Bijv.: Als bij aanroep van de routine de accu hex Ø9 bevat, zal de terugkeer naar het hoofdprogramma (en dus de tijdvertraging) 0,9 seconden duren.

Dit voorbeeld kan als volgt worden geprogrammeerd:
instructies:

A9 Ø9	LFA	#$Ø9;	laad de accu met hex Ø9
2Ø 1E F8	JSR	TØ1S:	roep vertragingsroutine aan instructies

= = INC16X = =

Deze routine incrementeert (= verhoogd met 1) een zestien bits getal. Het getal moet op pagina 0 staan en het adres in deze pagina moet in het X-register staan.

Voorbeeld:

Op adressen ØØ35 en ØØ36 staat het zestien bits getal 34FF, dat geïncrementeerd moet worden.

adres	data
ØØ35	FF
ØØ36	34

Na uitvoering van

formaat:

A2 35	LDX	#$35;	laad X-register met hex 35
20 21F8	JSR	INC 16X;	incrementeer 16 bit data

is de data: ØØ35 ØØ
ØØ36 35

Zoals bij de 6502-processor gebruikelijk staat het minst significante byte (het rechter byte) voorop, gevolgd door het meest significante byte (het linker byte).

Verandering in register(s): geen

6.1 Microcomputer MPS 65

= = DEC16X = =

Het was te verwachten, dat als er een incrementeer-routine is, dat er ook een decrementeer-routine bestaat. Decrementeren = met 1 verlagen. Deze routine werkt overeenkomstig INC16X.

Verandering in register(s): geen

= = ERR = =

Routine voor foutmelding met foutcode (bijv. ERROR 3).

De foutcode moet voor de aanroep van de subroutine in de accu worden gezet. Er zijn zestien foutcodes mogelijk: ØØ tot 0F.

Voorbeeld om op display ERROR 3 te zetten.

```
A9  Ø3      LDA   #$Ø3
2Ø  27 F8   TSR   ERR
```

N.B.: De subroutine ERR schakelt de displaymode op DISPSY.

Verandering in register(s): A, X

Code	hex toets	Code	funktietoets
18	0	51	G
28	1	52	S
38	2	62	I
48	3	54	L
14	4	64	B
24	5	58	A
34	6	68	E
44	7		
12	8		
22	9		
32	A		
42	B		
11	C		
21	D		
31	E		
41	F		

Tabel 4/6.1 -2:
De toetsencodes van de MPS 65

Adressen in pagina 0, die door de monitor worden gebruikt zijn te vinden in tabel 3.

6.1 Microcomputer MPS 65

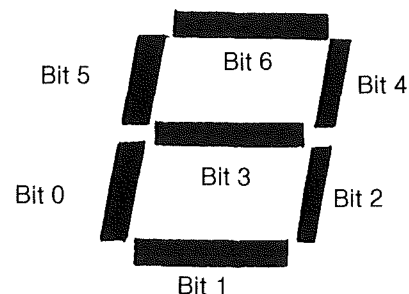
ØØAØ	= ADR16			ØØC3	= IRQF	
ØØA1	= ADR16+1			ØØC4	= IRQ65	(L)
ØØA2	= DAT8			ØØC5	= IRQ65	(H)
ØØA3	= DISP1					
ØØA4	= DISP2			ØØC6	= NMIF	
ØØA5	= DISP3			ØØC7	= NMI65	(L)
ØØA6	= DISP4			ØØC8	= NMI65	(H)
ØØA7	= DISP5					
ØØA8	= DISP6			ØØC9	= IRQMF	
				ØØCA	= IRQM	(L)
ØØA9	= STD			ØØCB	= IRQM	(H)
ØØAA	= MIN					
ØØAB	= SEK			ØØCC	= BRKUF	
				ØØCD	= BRKU	(L)
ØØAD	= GA			ØØCE	= BRKU	(H)
ØØAE	= GX					
ØØAF	= GY					
ØØCØ	= RESF					
ØØC1	= RES65	(L)				
ØØC2	= RES65	(H)				

Tabel 4/6.1 -3
De zero-page adressen.

Aansluitingen van de zevensegment displays:

MODE	8	8	8	8	8	8
DISPSY	DISP 1	DISP 2	DISP 3	DISP 4	DISP 5	DISP 6
DISPEX	ADR 16 + 1		ADR 16		DAT 8	

Figuur 4/6.1 -7
De monitoradressen van het display.



Figuur 4/6.1 -8
Displayaansturing.

6.1 Microcomputer MPS 65

Voorbeeld om ABCDEF op het display te zetten:

	bits	binair	hex.
A	0,2,3,4,5,6	0111 1101	7D
B	0,1,2,3,4,5,6	0111 1111	7F
C	0,1,5,6	0110 0011	63
D	0,1,2,4,5,6	0111 0111	77
E	0,1,3,5,6	0110 1011	6B
F	0,3,5,6	0110 1001	69

Door de gevonden hexadecimale getallen in de geheugenplaatsen DISP1 . . . DISP6 te zetten en dan de routine DISPEX aan te roepen, zullen de letters A . . . F op de zes displays verschijnen.

Interrupt afhandeling!

De afhandeling van optredende interrupts door de MPS 65.

Het systeem kent de volgende interrupts:

vectoradres	vector
RES65	reset vector
IRQ65	IRQ (hardware; maskable) vector
NMI65	NMI (hardware; non-maskable) vector
BRKU	breakvector (software interrupt) voor gebruiker (met monitorprogramma)
IRQM	IRQ-vector voor gebruiker (met monitorprogramma); wordt automatisch door de geprogrammeerde timer geactiveerd (elke 2,5 ms)

Bij gebruik van de vectoren door de gebruikers, moeten de volgende regels strikt in acht worden genomen:

1. Eerst het adres van het gebruikersinterruptprogramma in de vectorgeheugenplaatsen zetten.
2. De bij deze vector behorende vlag op nul zetten (d.w.z. de geheugenplaats die als vlag wordt gebruikt met de waarde 00 opvullen). Het vlagadres van elke vector is de geheugenplaats voor de betreffende vector.

6.1 Microcomputer MPS 65

De verschillende vectorafhandelingen **moeten** het volgende formaat hebben:

RES65
De vector wijst naar het aanvangsadres van het programma, dat gestart moet worden na indrukken van de RESET-toets.

```
IRQ65   .
        .
        (Programma)
        .
        JSR PLAXY
        RTI
```

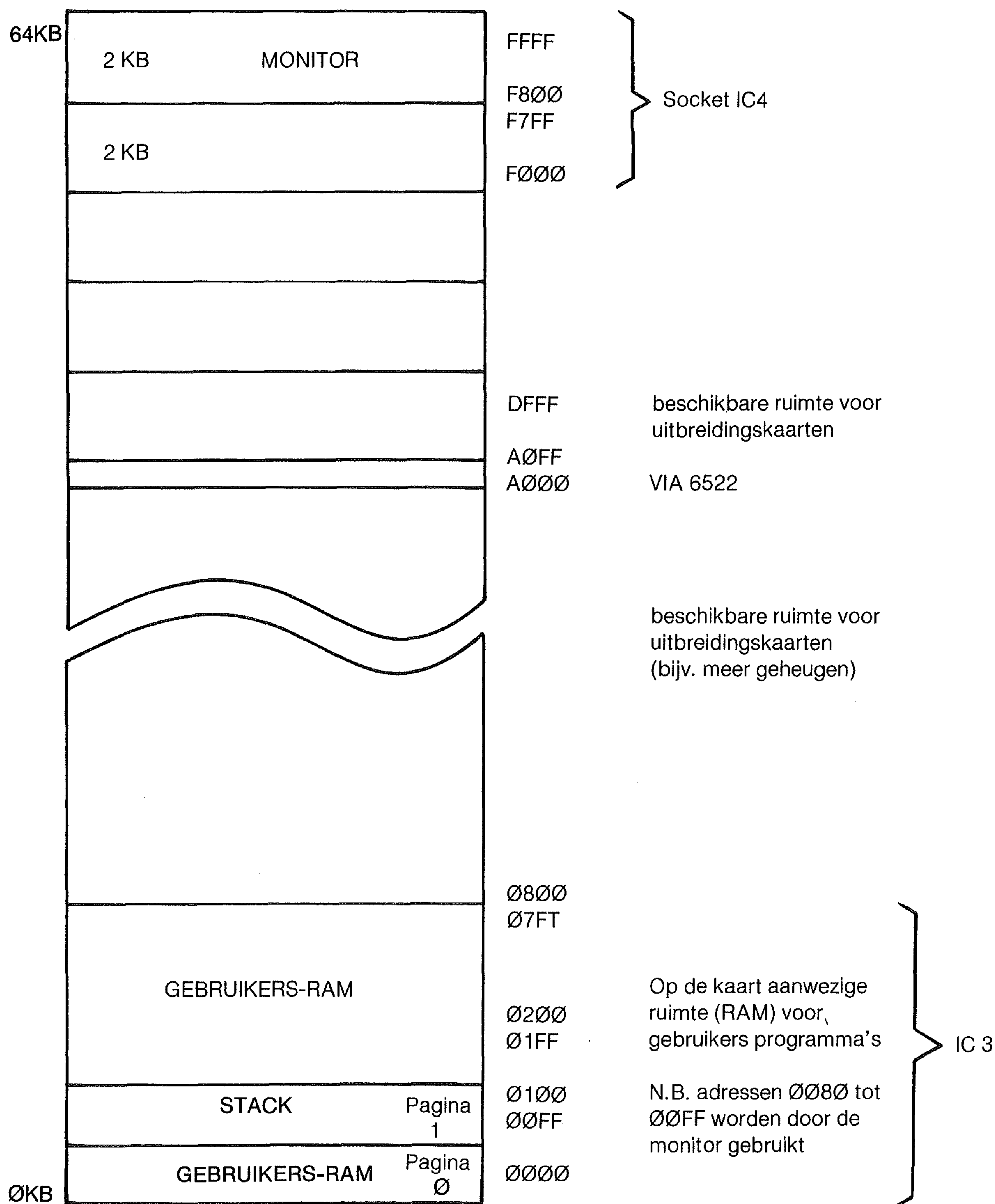
```
NMI65   .
        .
        (Programma)
        .
        JSR PLAXY
        RTI
```

```
BRKU    .
        .
        (Programma)
        .
        .
        JSR PLAXY
        RTI
```

```
IRQM    .
        .
        (Programma)
        .
        .
        RTS   !
```

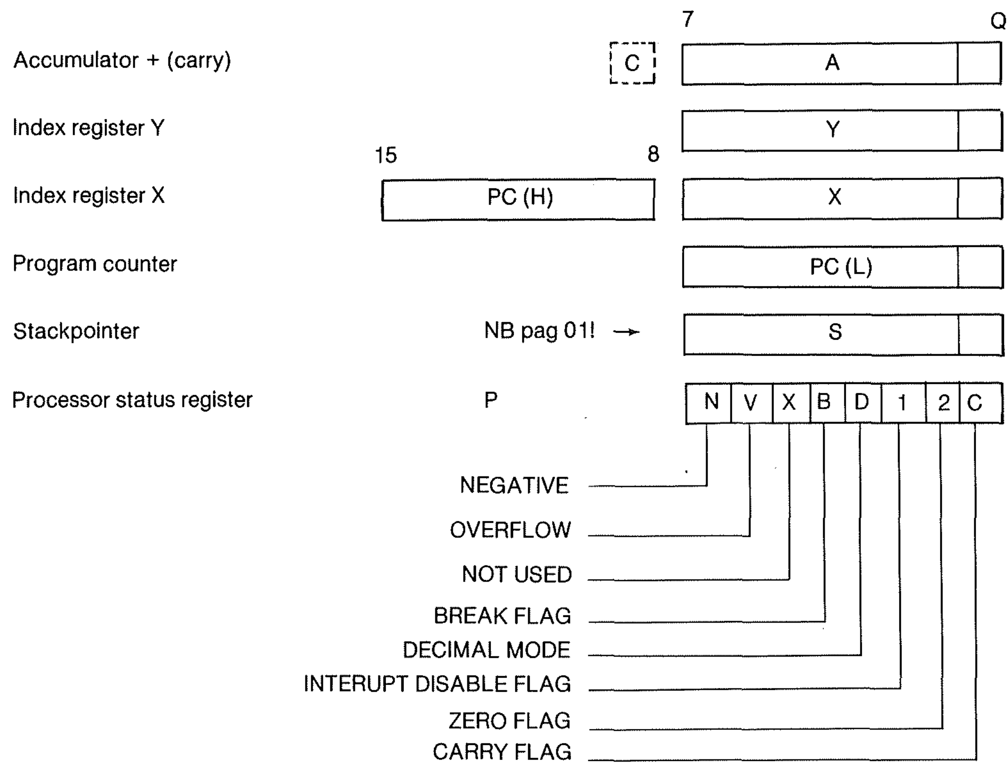
6.1 Microcomputer MPS 65

MPS 65 Geheugenindeling



Figuur 4/6.1 -9:
MPS 65 Geheugenindeling.

6.1 Microcomputer MPS 65



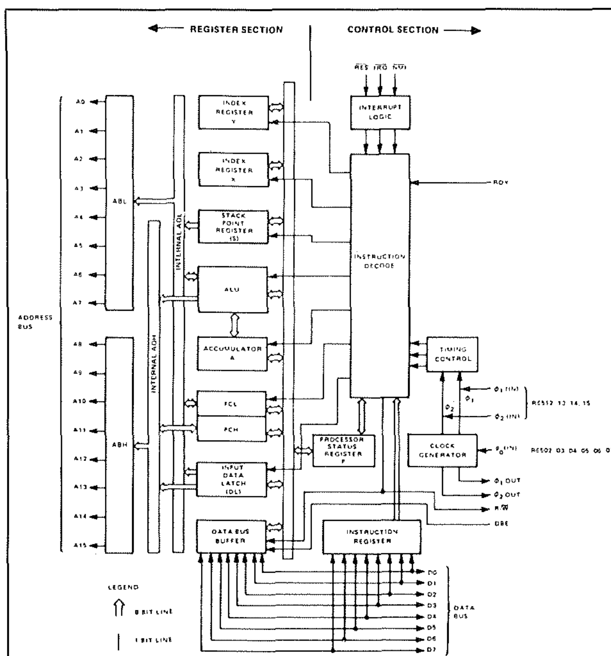
Figuur 4/6.1 -10:
De 6502 registers.

6.1 Microcomputer MPS 65

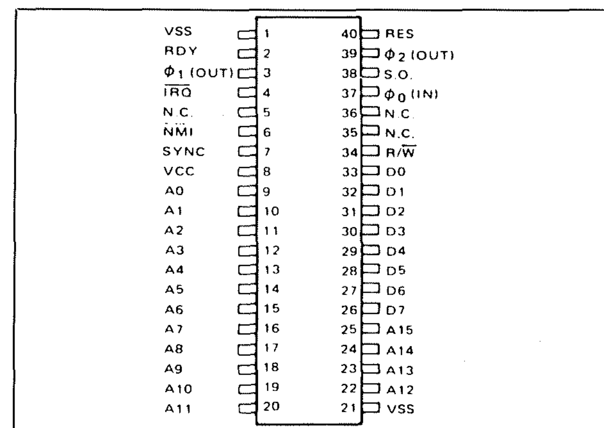
Systeemopbouw

De MPS 65 microcomputer is een op een Eurokaart (10 x 16 cm) gebouwde computer. Op de kaart bevinden zich alle voor de werking noodzakelijke IC's, alsmede een "operators console" in de vorm van een toetsenbord en een aantal displays.

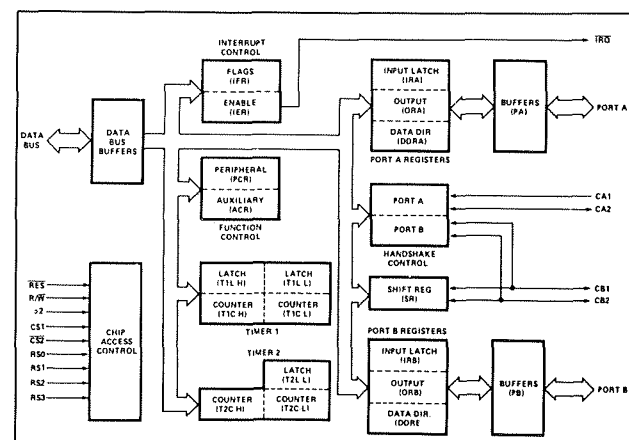
Als microprocessor is de 6502 CPU (central processing unit) van Rockwell gebruikt. Het toetsenbord en de display zijn met de computer verbonden via een VIA (versatile interface adaptor). Door middel van deze VIA is de CPU in staat het toetsenbord te lezen en het display te besturen. De VIA bevat nog andere logica, zoals programmeerbare timers, die door de CPU gebruikt kunnen worden.
De VIA is "familie" van de CPU. De op de MPS 65 gebruikte VIA is het type 6522 evenals de CPU van Rockwell.
De uitbreidingsconnector J1 is de systeembus. Hierop zijn de adres-, data- en stuurlijnen aangesloten (alsook de voedingsspanning(en).
IC4 is een 24 pens IC van het type 2716. Dit is een 2048 x 8 bit EPROM (Erasable programmable read only memory), waarin het monitorprogramma is opgeslagen.
IC3 is de 2KB ram van het type 2114 (6116). Het adresbereik omvat de adressen ØØØØ tot Ø7FF voor het systeem (de stack).



Figuur 4/6.1 -11:
Interne opbouw van de 6502 CPU.



Figuur 4/6.1 -12:
De aansluitingen van de 6502.



Figuur 4/6.1 -13:
Blokschema van de R6522 VIA.

6.1 Microcomputer MPS 65

De belangrijkste technische gegevens van de VIA zijn:

1. Twee 8 bits-bidirectionele ingangs/uitgangspoorten.
2. Twee 16 bits programmeerbare timers/counters.
3. Seriële ingangspoort.
4. Enkelvoudige voeding (+ 5V).
5. TTL-compatibel.
6. LMOS-compatibele controlelijnen.
7. 1 en 2 MHz klokfrequentie mogelijk.

6502 Adresseermogelijkheden

(de vetgedrukte data staat na uitvoering van de instructie in de accu)

IMMEDIATE (IMM)

LDA #$AA	0200	A9	Instructie
	0201	**AA**	data

ABSOLUTE (ABS)

LDA $0312	0200	AD	Instructie
	0201	12	adresbyte (laag)
	0202	03	adresbyte (hoog)
	0312	**AA**	data

ZERO PAGE (Z)

LDA (Z)-$BA	00BA	**AA**	data
	0200	A5	instructie
	0201	BA	adresbyte (laag!)

ABSOLUTE INDEXED (X) (ABS,X)

LDA, X	0200	BD	instructie
	0201	01	adresbyte (laag)
(stel X=5)	0202	03	adresbyte (hoog)
	0301		doeladres
		+X	offset
	0306	**AA**	data

ABSOLUTE INDEXED (Y) (ABS, Y)

als boven met Y-register in plaats van X-register

ZERO PAGE INDEXED (X) (Z, X)

	0020		
stel X=5			

6.1 Microcomputer MPS 65

		+X	offset
	0025	AA	data
LDA(Z), X	0200		instructie
	0201	20	adresbyte (laag!)

ZERO PAGE INDEXED (Y) (Z, Y)

als boven met Y-register in plaats van X-register

INDEXED, X INDIRECT (IND, X)

	0020		
		+X	offset
stel X=5	0025	12	adresbyte (laag)
	0026	03	adresbyte (hoog)
LDA ($20, X)	0200	A1	instructie
	0201	20	adresbyte (laag!)
	0312	AA	data

INDIRECT Y-INDEXED ((IND), Y)

stel Y=5	0020	12	adresbyte (laag)
	0021	03	adresbyte (hoog)
LDA ($20), Y	0200	B1	instructie
	0201	20	adresbyte (laag)
	0312	--	
		+Y	offset
	0317	AA	data

INDIRECT JUMP

JMP ($0212)	0200	6C	instructie
	0201	12	adresbyte (laag)
	0202	02	adresbyte (hoog)
	0212	02	adresbyte (laag)
	0213	03	adresbyte (hoog)
	0302		Na de uitvoering van de instructie gaat het programma vanaf geheugenplaats 0302 verder.

6.1 Microcomputer MPS 65

Memory referenced instructies

	IMM	ABS	Z	ABS, X	ABS, Y	Z, X	Z, Y	(IND,) X	(IND), Y	IND
ADC	69	6D	65	7D	79	75		61	71	
AND	29	2D	25	3D	39	35		21	31	
ASL		ØE	Ø6	1E		16				
BIT		2C	24							
CMP	C9	CD	C5	DD	D9	D5		C1	D1	
CPX	EØ	EC	E4							
CPY	CØ	CC	C4							
DEC		CE	C6	DE		D6				
EOR	49	4D	45	5D	59	55		41	51	
INC		EE	E6	FE		F6				
JMP		4C								6C
JSR		20								
LDA	A9	AD	A5	BD	B9	B5		A1	B1	
LDX	A2	AE	A6		BE		B6			
LDY	AØ	AC	A4	BC		B4				
LSR		4E	46	5E		56				
ORA	Ø9	ØD	Ø5	1D	19	15		Ø1	11	
ROL		2E	26	3E		36				
ROR		6E	66	7E		76				
SBC	E9	ED	E5	FD	F9	F5		E1	F1	
STA		8D	85	9D	99	95		81	91	
STX		8E	86				96			
STY		8C	84			94				

Implied instructies

				Instr.	code	vlag status
BRK	ØØ	PLP	28			
CLC	18	RTI	4Ø			
CLD	D8	RTS	6Ø	BCC	9Ø	C = Ø
CLI	58	SEC	38	BCS	BØ	C = 1
CLV	B8	SED	F8	BEQ	FØ	Z = 1
DEX	CA	SEI	78	BMI	3Ø	N = 1
DEY	88	TAX	AA	BNE	DØ	Z = Ø
INX	E8	TAY	A8	BPL	1Ø	N = Ø
INY	C8	TSX	BA	BVC	5Ø	V = Ø
NOP	EA	TXA	8A	BVS	7Ø	V = 1
PHA	48	TXS	9A			
PHP	Ø8	TYA	98	(Zie processor status register voor vlaggen)		
PLA	68					

Tabel 4/6.1 -4: 6502 Instructieset.

6.1 Microcomputer MPS 65

Programmavoorbeelden

Opdracht 1:

Op displays 1 en 5 moet het teken "L" worden weergegeven.

"L" is geen hexadecimaal teken, daarom kan dit alleen worden bereikt, door via externe aansturing de betreffende displaysegmenten te activeren. Dit gebeurt in de mode DISPEX, die normaal door het monitorprogramma wordt ingeschakeld wanneer naar een gebruikersprogramma wordt gesprongen.

Voor het teken "L" moeten de segmenten 0, 1 en 5 oplichten (zie fig. 8), hetgeen overeenkomt met hexadecimaal 23.

Om het monitorprogramma een "L" op display 1 en 5 te laten zetten, moet de code 23 op adressen DISP1 en DISP5 (ØØA3 en ØØA7) worden gezet.

Met een sprong naar MON65E wordt het gebruikersprogramma beëindigd.

Het programma:		adres:	hex-code:
LDA	#$23	0200	A9
		0201	23
STA	DISP1	0202	85
		0203	A3
STA	DISP5	0204	85
		0205	A7
JMP	MON65E	0206	4C
		0207	1B
		0208	F8

Voer het programma in (de hex-code) door elke hex-code in te voeren, gevolgd door de ENTER-toets.
Daarna met behulp van de ADRES-toets het startadres (0200) invoeren.

Het programma wordt gestart met de GO-toets.

Het display moet nu het volgende plaatje te zien geven:

L--- L-

Met behulp van de ENTER-toets kan men nu weer terugkeren naar de monitor.

Opdracht 2:

Op de displays moeten willekeurige segmenten aangestuurd kunnen worden. De codes daarvan staan in een tabel, TAB, die begint op geheugenplaats ØØØØ.

(De BPL spronggafstand wordt met behulp van de INDEX B-toets bepaald!)

Het programma:		adres:	hex-code:
	LDX #$05	0200	A2
		0201	05
HERH:	LDA TAB, X	0202	B5
		0203	00
	STA DISP1, X	0204	95
		0205	A3
	DEX	0206	CA
	BPL HERH	0207	10
		0208	F9 (VIA INDEX B)
	JMP MON65E	020A	4C
		020B	1B
		020C	F8

De zes tekencodes, die weergegeven moeten worden, worden in adressen ØØØØ-ØØØ5 gezet.

Bijvoorbeeld:

HALLO

6.1 Microcomputer MPS 65

TAB:	0000	3D ➧	00A3	DISP1
	0001	7D ➧	00A4	DISP2
	0002	23 ➧	00A5	DISP3
	0003	25 ➧	00A6	DISP4
	0004	77 ➧	00A7	DISP5
	0005	08 ➧	00A8	DISP6

Start het programma als bij opdracht 1.

Opdracht 3:

Toon een ingetoetste (hexadecimale) code op displays 5 en 6.

Aangezien de start van een gebruikersprogramma vanuit de monitor altijd het inschakelen van de DISPEX-mode tot gevolg heeft, is de eerste taak die het gebruikersprogramma moet uitvoeren, het terugzetten in de hexadecimale displaymode (DISPSY).

Het toetsenbord wordt met de monitorsubroutine KEYIN gelezen, de waarde van de accu (de toetswaarde) wordt dan in DATS gezet, waarna het monitorprogramma de code displayed.

Het programma:		adres:	hex-code:
	JSR DISPSY	0200	20
		0201	09
		0202	F8
HERH:	JSR KEYIN	0203	20
		0204	00
		0205	F8
	STA DATS	0206	85
		0207	A2
	JMP HERH	0208	4C
		0209	03
		020A	02

Start het programma als bij opdracht 1. Dit programma zal telkens na indrukken van een toets de toetswaarde in displays 5 en 6 tonen, behalve van de RESET-toets (zie tabel 2). De RESET-toets is dan ook de enige methode om het programma te onderbreken en terug te komen naar de monitor.

Opdracht 4:

Verhoog de waarde op display 1 en 2 voortdurend met 1 (incrementeren)

Er wordt bij dit programma gebruik gemaakt van de tijdvertragingsroutine, om de diverse waarden op de display ook zichtbaar te maken.

Het programma:		adres:	hex-code
	JSR DISPSY	0200	20
		0201	09
		0202	F8
HERH:	LDA #$03	0203	A9
		0204	03 ; 3x
	JSR TØ1S	0205	20 ; 0,1 sec.
		0206	1E
		0207	F8
	INC A1	0208	E6 ; Increment
		0209	A1 ; DATS
	JMP HERH	020A	4C
		020B	03
		020C	02

Opdracht 5:

Start een programma op een bepaalde tevoren ingegeven tijd.

Een singleboard computer als de MPS 65 kan eenvoudig worden uitgebreid met elektronica om er een schakelklok van te maken.
Bij deze opdracht wordt aangetoond hoe het programma, met gebruikmaking van de interne, onder de monitor lopende klok een ander in het geheugen aanwezig programma kan starten.

6.1 Microcomputer MPS 65

Er wordt vanuitgegaan, dat de schakeltijd in de geheugenplaatsen ØØ, Ø1 en Ø2 van pagina nul staan. Van de lopende klok staat de tijd in de geheugenplaatsen STD, MIN, SEC op de adressen A9, AA en AB van pagina nul.

Het programma:	adres:	hex-code:
LDX #$02	0200	A2
	0201	02
LDA 00, X	0202	B5 ; haal schakeltijd op
	0203	00
CMP STD, X	0204	D5 ; vergelijk met kloktijd
	0205	A9
BNE $200	0206	D0 ; opnieuw indien niet
	0207	F8 ; gelijk
DEX	0208	CA ; nu min en uren
BPL $202	0209	10
	020A	F7
	020B	; hier begint het te starten programma.

Het programma dat in het geheugen staat vanaf adres 020B zal worden uitgevoerd, zodra de lopende klok in adressen A9 t/m AB gelijk is aan de schakeltijd in de adressen ØØ t/m Ø1. Met behulp van dit programma kunnen dan de gewenste functies worden uitgevoerd.

Het starten van programma's met gedefinieerde registers!

Wanneer een gebruikersprogramma gestart wordt met de GO-toets, worden door het monitorprogramma de inhoud van de geheugenplaatsen GA, GX en GY gekopieerd in respectievelijk de accu, het X- en het Y-register. Pas daarna wordt het gebruikersprogramma gestart.
Normaal zal men een gebruikersprogramma altijd zelf initialiseren. Deze functie is echter van belang, als men bepaalde delen van het programma wil testen, waarbij dan de inhoud van de accu en de registers een vorig resultaat voorstellen. Dit "vorig" resultaat kan men dan ingeven in de genoemde geheugenplaatsen (ØØAD, ØØAE en ØØAF).

6.1 Microcomputer MPS 65

Mini-cursus MPS 65 computersysteem

Computers werken in een binair (= tweetallig) getalstelsel. Hierbij is het grondtal 2.
Bij het normale decimale stelsel is het grondtal tien. Een getal is opgebouwd uit een aantal cijfers, die afhankelijk van hun plaats aangeven hoe vaak een bepaalde macht van 10 in het totale getal voorkomt.

Dus 1984 wil zeggen:

$4 \times 10^0 = 4 \times 1$
$8 \times 10^1 = 8 \times 10$
$9 \times 10^2 = 9 \times 100$ en
$1 \times 10^3 = 1 \times 1000$

Omdat het grondtal 10 is, zijn er ook 10 cijfers (0 t/m 9) om aan te geven hoe vaak een bepaalde macht voorkomt.

In het binaire stelsel zijn er slechts twee cijfers (0 en 1) om aan te geven hoe vaak een bepaalde macht van 2 voorkomt.

De binaire getal 1011 wil zeggen:

$1 \times 2^0 = 1 \times 1 = 1$
$1 \times 2^1 = 1 \times 2 = 2$
$0 \times 2^2 = 0 \times 4 = 0$
$1 \times 2^3 = 1 \times 8 = 8$
+
11

Het binaire getal 1011 komt dus overeen met het decimale getal 11.

Voor computers is het binaire stelsel uitermate geschikt, daar de twee getallen 0 en 1 op eenvoudige wijze in elektrische toestanden kan worden omgezet (aan/uit; 0 Volt/+5 V; stroom/geen stroom; e.d.).

Voor de programmeur echter is het binaire stelsel wat minder eenvoudig omdat het erg lange getallen oplevert, die bovendien, omdat er alleen nullen en enen in voorkomen, erg onoverzichtelijk zijn.
Bijvoorbeeld: decimaal 1984 is binair 11111000000.

Elk cijfer in het binaire stelsel heet een bit. Door nu 4 bits te combineren, ontstaat een getal, dat nibble genoemd wordt. Een nibble kan $2^4 = 16$ verschillende waarden aannemen. Deze worden weergegeven door de cijfers 0 t/m 9 en de letters A t/m F.

Een getal voorgesteld door meerdere nibbles heeft als grondtal 16. We noemen zo'n getal hexadecimaal.

Dus 07C0 hexadecimaal is:

$0 \times 16^0 = 0 \times 1 = 0$
$C \times 16^1 = 12 \times 16 = 192$
$7 \times 16^2 = 7 \times 256 = 1792$
$0 \times 16^3 = 0 \times 4096 = 0$
+
1984 decimaal

Om aan te geven dat een getal hexadecimaal is wordt het vaak gevolgd door de letter H. Dus 07C0 H.

Voor de meeste computers bestaan assemblers, die de mnemonic-code van instructies en de daarbij behorende data en adressen omzetten in machinecode (hexadecimale/binaire getallen). Deze assemblers gebruiken vaak het $-teken om aan te geven dat een getal hexadexi-

6.1 Microcomputer MPS 65

maal is. Dit $-teken wordt dan voor het getal geplaatst.

Veel van de populaire microprocessoren werken met een databus, die 2 nibbles (= 8 bits) breed is. Twee nibbles worden een byte genoemd.

Voor de volledigheid zij nog vermeld, dat 2 bytes (= 16 bits) een woord wordt genoemd en 2 woorden (= 4 bytes = 32 bits) een dubbel woord of longword.

De volgende tabel verduidelijkt de samenhang tussen decimale, binaire en hexadeximale getallen.

decimaal	binair	hexadecimaal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Nu volgen een aantal voorbeelden van werken met hexadecimale en binaire getallen.

Voorbeeld 1:

AND-functie met 4 bits.

binair	hexadecimaal
0101	$5
∧0110	∧$6
0100	$4

Voorbeeld 2:

AND-functie met 8 bits.

binair	hexadecimaal
1101.1100	$DC
∧1100.0111	∧$C7
1100.0100	$C4

Voorbeeld 3:

ORA-functie (OR).

binair	hexadecimaal
0111.0000	$70
∨0011.0100	∨$34
0111.0100	$74

Voorbeeld 4:

EOR-functie (exclusieve OR).

binair	hexadecimaal
0101.0000	$50
∨0011.0100	⊻$34
0110.0100	$64

Voorbeeld 5:

ADD-functie (optellen)

binair	hexadecimaal
0101.0010	$52
+0101.1100	+$5C
1010.1110	$AE

Voorbeeld 6:

ADC-functie (optellen met "carry").

binair	hexadecimaal
0101.0010	$ 52
+ 1101.0110	+ $ D6
1.0010.1000	$1.28



Carry bit (1 onthouden)

6.1 Microcomputer MPS 65

Het carry bit geeft een overdracht aan naar het negende bit. D.w.z. voor overdracht naar een eventuele optelling van een meer significant deel van de op te tellen getallen.

N.B. De 6502 microprocessor heeft een mode, waarbij decimaal wordt gerekend. In de binaire mode werkt de carry-functie als in voorbeeld 6. D.w.z. dat het carry bit wordt "geset" (logisch "1" wordt) als de waarde $FF wordt overschreden. Bij decimaal rekenen gebeurt dit echter als de waarde $99 overschreden wordt. De mode verandert met de instructies SED (set Decimal) en CLD (clear Decimal). Het is zaak bij een gebruikersprogramma zeker te zijn van de mode waarin men werkt. Het is dan ook een goede gewoonte om een programma te beginnen met CLD en die delen van het programma, waarin bewust decimaal gerekend moet worden te beginnen met SED en te beëindigen met CLD.

CPU-registers.

De CPU (central processing unit) 6502 heeft een aantal registers, waarin bewerkingen kunnen worden uitgevoerd. Een van de registers is de accumulator, kortweg accu genoemd. Deze accumulator heeft een breedte van 8 bits (1 byte), daarnaast is er nog een carry-bit, die door de accu vaak als negende bit bij bewerkingen wordt gebruikt.

Om bewerkingen op twee getallen uit te voeren, moet er altijd één van de twee in de accu aanwezig (geladen) zijn. Het resultaat van de bewerking staat na de bewerking in de accu.

Voorbeeld 3 als programma.

Om de ORA-functie van voorbeeld 3 uit te voeren moeten de volgende acties ondernomen worden:

1. Laad de accu met het hexadecimale getal 70.
2. Voer een OR-functie uit tussen het getal in de accu en een getal op een geheugenplaats.
3. Het resultaat staat in de accu.

Deze beschrijving voldoet wel voor de programmeur, maar niet voor de computer. Deze heeft binaire code nodig. Door gebruik te maken van mnemonics (afkortingen) is het mogelijk het programma compact te schrijven, daarna kan het eenvoudig worden gecodeerd in binaire (of hexadecimale) code (handmatig of met behulp van een assembleerprogramma). De fabrikant heeft voor de CPU mnemonics en de daarbij behorende codes vastgelegd.
In mnemonic-vorm zou het programma uit voorbeeld 3 er als volgt kunnen uitzien:

```
LDA  #$70
ORA  #$34
BRK
```

Hierbij zijn LDA, ORA en BRK instructies, eventueel gevolgd door data.

- # is een assembler-code, die immediate adressing aangeeft, d.w.z. de te bewerken data staat op de geheugenplaats die volgt op de geheugenplaats van de instructie zelf.
- $ is het teken dat aangeeft dat het hier om een hexadecimaal getal gaat.

Zoeken we in de 6502 instructietabel de bijbehorende codes op, dan kunnen we

6.1 Microcomputer MPS 65

van het programma de volgende "listing" maken:

adres	hex-code	instructie	operand
0200	A9	LDA	#$70
0201	70		
0202	09	ORA	#$34
0203	34		
0204	00	BRK	

Als dit programma in de MPS 65 wordt ingevoerd en daarna op adres 0200 gestart, worden de drie instructies afgewerkt.
De BRK-instructie zorgt voor een terugkeer naar het monitorprogramma, waarbij de monitor de inhoud van X-register, Y-register en de accu op respectievelijk displays 1 en 2, 3 en 4, 5 en 6 laat zien.

Displays:

(X-Reg.)	(Y-Reg.)	(Accu)
0 0	0 0	7 4

Voorbeeld 5 als programma

Om zeker te zijn van het juiste resultaat bij voorbeeld 5, moet voorafgaand aan de optelling in het programma het carry-bit nul gemaakt worden, er is immers nog geen sprake van een overdracht van een vorig resultaat.

Met behulp van de 6502 instructietabel schrijven we het programma en de erbij behorende code:

adres	hex-code	instructie	operand
0200	A9	LDA	#$52
0201	52		
0202	18	CLC	
0203	69	ADC	#$5C
0204	5C		
0205	00	BRK	

Na invoering van dit programma staat het resultaat van de optelling in de displays 5 en 6 (accu-inhoud).

In de tot nu toe gebruikte voorbeelden stonden de getallen die als factoren in de bewerking werden gebruikt in het programma zelf (onmiddellijk na de instructie).
Deze vorm van adresseren van de factoren (operands in computertaal) noemt men immediate adressering.
Echter de 6502 CPU kent een veelheid van adresseringsmogelijkheden (zie ook tabel 4).

In het voorafgaande voorbeeld is nog een tweede adresseringsvorm aanwezig, de zogenaamde impliciete adressering, namelijk de CLC-instructie (clear carry = maak het carry-bit 0). De instructie impliceert, waarop deze betrekking heeft. Deze instructies bestaan dan ook slechts uit één byte. Andere voorbeelden zijn CLD, TAX, PHA, PLA etc.

Een andere adresseermogelijkheid is die, waarbij de instructie wordt gevolgd door het adres van de geheugenplaats, waar de data zich bevindt. Deze wijze van adresseren staat bekend als absolute adressering. De instructie wordt gevolgd door twee bytes, het eerste daarvan bevat het minst significante byte van het adres, het daaropvolgende bevat het meest significante byte.

De opdracht: laad de accu met de inhoud van geheugenplaats 0300 wordt in mnemonic LDA $0300 en in hex-code:

AD	Instructie
00	adresbyte (laag)
03	adresbyte (hoog)

6.1 Microcomputer MPS 65

Voorbeeld 7:

De inhoud van geheugenplaatsen 0300 en 0301 moet 'ge-OR-ed" worden. Het resultaat van deze OR-functie moet worden opgeborgen in geheugenplaats 0302. Het programma zelf begint op geheugenplaats #0200.

adres	hex-code	instructie	operand
0200	AD	LDA	$0300
0201	00	16 bits adres	
0202	03		
0203	2D	AND	$0301
0204	01	16 bits adres	
0205	03		
0206	8D	STA	$0302
0207	02	16 bits adres	
0208	03		
0209	00	BRK	

Als we in ons voorbeeld met de OR-functie de volgende getallen willen bewerken:

```
  0110.1001        $69
  0011.1100        $3C
∧                ∧
  ---------        ---
  0010.1000        $28
```

dan moeten we met behulp van de ADDRES-toets en de ENTER-toets de waarden $69 en $3C invoeren in respectievelijk de geheugenplaatsen $0300 en $0301.

Wederom met behulp van de ADDRES-toets en vervolgens de GO-toets wordt het programma op adres $0200 gestart. Na afloop zal de waarde $28 in geheugenplaats $0302 staan.

ZERO-Page adressering

De eerste 256 bytes van het geheugen (de adressen ØØØØ-ØØFF) worden pagina nul of zero-page genoemd.
Adressen op pagina nul kunnen natuurlijk via de eerder besproken absolute adressering worden bereikt.
Echter de 6502 CPU kent een speciale instructieset, voor instructies, die aan pagina 0 refereren. Het voordeel van deze instructies is, dat zij slechts twee bytes geheugenruimte innemen en dat zij sneller uitgevoerd worden.
De instructie LDA #$0010
(in hex-code AD 1000)
kan ook worden geschreven als zero-page-instructie
LDA $10 (in hex-code A 5 # 10)

Voorbeeld 8:

Als voorbeeld zeven, echter in plaats van de adressen 0300, 0301 en 0302 worden nu de adressen 0010, 0011 en 0012 gebruikt.

adres	hex-code	instructie	operand adres
0200	A5	LDA	$10
0201	10	(8 bit adres)	
0202	25	AND	$11
0203	11	(8 bit adres)	
0204	85	STA	$12
0205	12	(8 bit adres)	
0206	00	BRK	

Geïndexeerde adressering.

Bij geïndexeerde adressering wordt de inhoud van X- of Y-register mede gebruikt bij de bepaling van het operand adres.

6.1 Microcomputer MPS 65

Niet alle instructies hebben zowel X- als Y-indexering.

Voorbeeld 9:

Kopieer de inhoud van geheugenplaatsen Ø2ØØ-Ø2ØØ naar Ø3ØØ-Ø3ØD

adres	hex-code	instructie
0200	A2	LDX #$00
0201	00	
0202	BD	LDA $0200, X
0203	00	
0204	02	
0205	9D	STA $0300, X
0206	00	
0207	03	
0208	E8	INX
0209	E0	CPX #0E
020A	0E	
020B	D0	BNE $0202
020C	F5	
020D	00	BRK

Na de instructie LDX $00 is de inhoud van het X-register 00.
De instructies LDA $0200 en STA $0300 hebben betrekking op de adressen 0200 respectievelijk 0300.
Na uitvoering van de instructie INX is de inhoud van het X-register 01 geworden. De CPX #$0E-instructie vergelijkt de inhoud van het X-register met de waarde $0E. Aangezien dit ongelijk is zal de daaropvolgende BNE-instructie de computer terugvoeren naar adres 0202. De LDA- en STA-instructies worden opnieuw uitgevoerd, echter nu op adressen 0201 respectievelijk 0301 (0200 + X-register = 0201 en 0300 + X-register = 0301). Dit gaat door, totdat het X-register de waarde $0E bereikt, dan wordt niet meer teruggesprongen naar adres 0202 aangezien de CPX-instructie in een "equal" resulteert (Zero-flag wordt "1"). De laatste gekopieerde geheugenplaats is dan 020D (0200 + X-register = 0200 + 0D = 020D).

Ook de geïndexeerde adressering kent zero-page-instructies. Bij het bestuderen van de 6502 instructie-tabel valt op, dat er hier slechts twee Y-indexeringsinstructies zijn, namelijk die instructies die op het X-register zelf werken en waardoor dus uiteraard het X-register zelf niet gebruikt kan worden, als indexeringsregister.

De indexed X indirect adressering.

Eerst indexeren, dan indirect. Deze adressering is een vorm van zero-page adressering.
De instructiecode wordt gevolgd door een adres op pagina nul. Bij dit adres op pagina nul wordt de inhoud van het X-register opgeteld. De data die op het op deze wijze ontstane adres staat, vormt samen met de data van de volgende geheugenplaats het absolute adres waar de operand staat. Dit klinkt ingewikkeld, het volgende voorbeeld dient ter verduidelijking.

Voorbeeld 10:

adres	hex-code	instructie + operand (adres)
0200	A2	LDX 02
0201	02	
0202	A1	LDA ($20, X)
0203	20	
0204	00	BRK
0205	EA	NOP
0020	--	
0021	--	
0022	05	
0023	02	

6.1 Microcomputer MPS 65

Bij uitvoering van dit programma zal de computer de volgende stappen ondernemen bij de instructie LDA.

Bij adres 0020 wordt de inhoud van het X-register opgeteld, dit levert op $0020 + $02 = $0022. Nu wordt gekeken welk adres op geheugenplaatsen 0022 en 0023 is opgeslagen. Dit blijkt adres 0205 te zijn. Nu wordt de data die in geheugenplaats $0205 staat in de accu geladen. Dit is dus $EA.

De indirect, Y indexed adressering.

Eerst indirect, dan indexeren.

Aan de hand van het voorbeeld zal ook deze adresseringsvorm duidelijk worden.

Voorbeeld 11:

adres	hex-code	instructie + operand (adres)
0200	A0	LDY $05
0201	05	
0202	B1	LDA ($20), Y
0203	20	
0204	00	BRK
0205	EA	NOP
0020	00	
0021	02	

Bij dit voorbeeld voert de computer de volgende handelingen uit bij de instructie LDA.

Er wordt gekeken welk 16 bits adres op de geheugenplaatsen 0020 en 0021 staat. Dit blijkt te zijn 0200. Hierbij wordt de inhoud van het X-register opgeteld. Dit levert op $0200 + $05 = $0205. Nu wordt de inhoud van geheugenplaats 0205 in de accu geladen. Dit is dus $EA.

6.1 Microcomputer MPS 65

ADC	Add Memory to Accumulator with Carry	JSR	Jump to New Location Saving Return Address
AND	"AND" Memory with Accumulator		
ASL	Shift Left One Bit (Memory or Accumulator)	LDA	Load Accumulator with Memory
		LDX	Load Index X with Memory
BCC	Branch on Carry Clear	LDY	Load Index Y with Memory
BCS	Branch on Carry Set	LSR	Shift Right One Bit (Memory or Accumulator)
BEQ	Branch on Result Zero		
BIT	Test Bits in Memory with Accumulator	NOP	No Operation
BMI	Branch on Result Minus	ORA	"OR" Memory with Accumulator
BNE	Branch on Result not Zero		
BPL	Branch on Result Plus	PHA	Push Accumulator on Stack
BRK	Force Break	PHP	Push Processor Status on Stack
BVC	Branch on Overflow Clear	PLA	Pull Accumulator from Stack
BVS	Branch on Overflow Set	PLP	Pull Processor Status from Stack
CLC	Clear Carry Flag	ROL	Rotate One Bit Left (Memory or Accumulator)
CLD	Clear Decimal Mode	ROR	Rotate One Bit Right (Memory or Accumulator)
CLI	Clear Interrupt Disable Bit	RTI	Return from Interrupt
CLV	Clear Overflow Flag	RTS	Return from Subroutine
CMP	Compare Memory and Accumulator		
CPX	Compare Memory and Index X	SBC	Subtract Memory from Accumulator with Borrow
CPY	Compare Memory and Index Y	SEC	Set Carry Flag
		SED	Set Decimal Mode
DEC	Decrement Memory by One	SEI	Set Interrupt Disable Status
DEX	Decrement Index X by One	STA	Store Accumulator in Memory
DEY	Decrement Index Y by One	STX	Store Index X in Memory
		STY	Store Index Y in Memory
EOR	"Exclusive-Or" Memory with Accumulator		
		TAX	Transfer Accumulator to Index X
INC	Increment Memory by One	TAY	Transfer Accumulator to Index Y
INX	Increment Index X by One	TSX	Transfer Stack Pointer to Index X
INY	Increment Index Y by One	TXA	Transfer Index X to Accumulator
		TXS	Transfer Index X to Stack Pointer
JMP	Jump to New Location	TYA	Transfer Index Y to Accumulator

Tabel 4/6.1 -5: De 6502 instructies op alfabetische volgorde

6.1 Microcomputer MPS 65

Aansluitingen van de uitbreidingsconnector J1

De uitbreidingsconnector bevat de aansluitingen van de uitgebreide SMP-bus.

Aan-sluit-punt	signaal	aan-sluit-punt	signaal
1	−15V	1	− 12V
2	− 5V	2	GND
3	s.o.	3	+ 5V
4	Ø2	4	—
5	R/W	5	A12
6	$\overline{\text{RES}}$	6	AØ
7	SYNC	7	A13
8	$\overline{\text{R/W}}$	8	A1
9	—	9	A14
10	RAM R/W	10	A2
11	—	11	A15
12	READY	12	A3
13	BUSEN	13	—
14	DØ	14	A4
15	—	15	—
16	D1	16	A5
17	—	17	—
18	D2	18	A6
19	—	19	—
20	D3	20	
21	D3	21	A7
22	—	22	—
23	D4	23	A8
24	$\overline{\text{IRQ}}$	24	—
25	D5	25	A9
26	—	26	—
27	D6	27	A10
28	—	28	—
29	D7	29	A11
30	—	30	—
31	—	31	—
32	+ 15V	32	GND
	+ 5V		+ 12V

Tabel 4/6.1 -6:
Uitgebreide SMP-Bus aansluitingen.

De standaard adres- en data-lijnen, de stuurbussignalen en de voedingsspanningen zijn hierbij op gestandaardiseerde punten aangesloten, zodat uitbreidingskaarten, die dezelfde busstructuur hebben eenvoudig via een moederboard of een bedrade groep connectoren met de MPS 65 worden verbonden. De busconnector is een 64-polige DIN connector. Rij b is niet gebruikt. Zie tabel 6.
Een streepje betekent, dat de betreffende pen niet is aangesloten.

6.1 Microcomputer MPS 65

Het monitorprogramma van de MPS 65.

```
;### ZERO-PAGE ###

;*******************
;* ANWENDER-BEREICH*
;*******************
*=$00A0
ADR16 *=*+2
DAT8  *=*+1

*=$00A3
DISP1 *=*+1
DISP2 *=*+1
DISP3 *=*+1
DISP4 *=*+1
DISP5 *=*+1
DISP6 *=*+1

*=$00A9
STD   *=*+1
MIN   *=*+1
SEK   *=*+1
T004  *=*+1

*=$00AD
GA    *=*+1
GX    *=*+1
GY    *=*+1

*=$00C0
RESF  *=*+1
RES65 *=*+2

*=$00C3
IRQF  *=*+1
IRQ65 *=*+2

*=$00C6
NMIF  *=*+1
NMI65 *=*+2

*=$00C9
IRQMF *=*+1
IRQM  *=*+2

*=$00CC
BRKUF *=*+1
BRKU  *=*+2

*=$00FD
PAXY  *=*+3 ;P/AXY


;*******************
;*  SYSTEMBEREICH  *
;*******************
*=$0080
REC

*=$0100
BUF
*=$00D3
KALT  *=*+2
UHRR  *=*+2

*=*+12 ;RECORDER
;
KEY   *=*+1
FGPB  *=*+1
FGHEX *=*+1
FGENT *=*+1

R0    *=*+1
R1    *=*+1
R2    *=*+1
R3    *=*+1

RKX   *=*+1
RK1   *=*+1
RI1   *=*+1
;
;## END ZERO-PAGE ##
```

```
B      =$F600
ARBRAM=$0200
ZEIT1 =2501-2
;
V      =$A000
PORTB =V
PORTA =V+1
DIRPB =V+2
DIRPA =V+3
LATC1L=V+4
LATC1H=V+5
TIM1L =V+6
TIM1H =V+7
TIM2L =V+8
TIM2H =V+9
SR     =V+$A
ACR    =V+$B
PCR    =V+$C
IFR    =V+$D
IER    =V+$E

TG=$51
TS=$52
TI=$62
TL=$54
TB=$64
TA=$58
TE=$68

*=$0200
ANFMON
;-SYSTEMEINSPRUENGE-
JMP KEYIN+B
JMP KEYIN1+B
JMP DISPEX+B
JMP DISPSY+B
JMP PHAXY+B
JMP PLAXY+B
JMP ASK2+B
JMP ASK4+B
JMP MON65+B
JMP MON65E+B
JMP T01S+B
JMP INC16X+B
JMP DEC16X+B
JMP ERR+B
RTS
.BYT 0,0
JMP STRICH+B
JMP SUCHEX+B

;===================
.BYT $26
.BYT '(C)BY THALER'
.BYT '10/1983'
;===================

;#### PROGRAMME ####

ERR ;ERROR-<A> (0-F)
  JSR STRICH+B
  AND #$0F
  TAY
  LDX #5
  LDA DISTAB+B,Y
ER1
  STA DISP1,X
  LDA TER+B-1,X
  DEX
  BPL ER1
  JSR KEYINE+B
  JSR DISPSY+B
  RTS
;
;
INDEXB ;ABS.=>REL.AD
  LDX #ADR16;BRANCH?
  JSR DEC16X+B
  LDX #0
  LDA (ADR16,X)
  LDX #ADR16
  JSR INC16X+B
  LDX #7
BTAB1
  CMP BTAB+B,X
  BEQ BOK
  DEX
  BPL BTAB1
  LDA #4 ;ERROR4
  JSR ERR+B
  JMP WARM1+B
```

```
BOK
  LDA ADR16 ;SICHERN
  PHA
  LDA ADR16+1
  PHA

  JSR STRICH+B
  LDA #$2F ;B
  STA DISP5
  LDA #9 ;R
  STA DISP6
  JSR ASK4+B

  PLA
  STA R2
  PLA
  STA R1

  LDA ADR16;REL.SPR.
  SEC
  SBC R1
  STA R3;LH
  LDA ADR16+1
  SBC R2 ;HB

  BMI BNEG
  BNE BERR2
  DEC R3 ;POS.
  LDA R3
  CMP #128
  BCS BERR2;>=
  LDX #0
  STA (R1,X);REL.ADR
  JMP BEND+B

BNEG
  CMP #$FF
  BNE BERR3
  DEC R3
  LDA R3
  BPL BERR3
  LDX #0
  STA (R1,X)

BEND
  LDA R1
  STA ADR16
  LDA R2
  STA ADR16+1
  JMP WARM1+B
BERR2
  LDA #3
  BNE BERR4
BERR3
  LDA #2
BERR4
  JSR ERR+B
  JMP BEND+B
BTAB
.BYT $90,$B0,$F0,$30
,$D0,$10,$50,$70

INDEXC ;SHIFT
  LDX #$FF
IC1
  DEX
  LDA $200,X
  STA $201,X
  LDA #$EA
  STA $200,X
  CPX ADR16
  BNE IC1
  INC ADR16
  JMP WARM1+B
;
PHAXY
  CLC
  BCC PLAXY+1
PLAXY
  SEC
  STA PAXY+2
  PLA
  STA PAXY
  PLA
  STA PAXY+1
  BCS P1
  LDA PAXY+2
  PHA
  TXA
  PHA
  TYA
```

```
  PHA
  LDA PAXY+2
  BCC P2
P1
  PLA
  TAY
  PLA
  TAX
  PLA
P2
  INC PAXY
  BNE *+4
  INC PAXY+1
  JMP (PAXY)
;
ASCII ;(A)=>(A).(X)
  PHA
  JSR ASC1+B
  TAX
  PLA
  ROR A
  ROR A
  ROR A
  ROR A
  JSR ASC1+B
  RTS
ASC1
  AND #$0F
  CMP #$0A
  BCS ASC2
  ORA #$30
  RTS
ASC2
  ADC #$36
  RTS
;
;##### ZEITEN ######
T01S ;<A>*0,1S
  JSR PHAXY+B
  LDY #75
  LDX #255
  DEX
  BNE *-1
  DEY
  BNE T01S+5
  SEC
  SBC #1
  BNE T01S+3
  JSR PLAXY+B
   RTS
;
;*******************
;*TASTATUR, ANZEIGE*
;*INITIALISIERUNGEN*
;*******************

MON65E ;NACH ENTER
  JSR KEYINE+B

MON65
  LDX #$FF
  TXS
  JSR INIT1+B ;VIA
  LDA #$33
  CMP KALT
  BNE INITK
  CMP KALT+1
  BEQ WARMST
INITK
  STA KALT
  STA KALT+1
  JSR INIT2+B
  JSR INIT4+B

;MPS-65
  JSR DISPEX+B
  LDX #5
D65
  LDA TAB65+B,X
  STA DISP1,X
  DEX
  BPL D65

  JSR KEYINE+B ;ENT?

  JSR DISPSY+B
```

6.1 Microcomputer MPS 65

```
;SELBSTSTART?
  DEC $E003 ;01?
  BNE WARMST
  INC $E004 ;FF?
  BNE WARMST
  JSR STRICH+B
  JMP $E005

WARMST
  LDA RESF
  BNE *+5
  JMP (RES65)
  JSR DISPSY+B
  LDA #<ARBRAM
  STA ADR16
  LDA #>ARBRAM
  STA ADR16+1
WARM1
  LDX #0
  LDA (ADR16,X)
  STA DAT8
TASTEN
  JSR KEYIN+B
  CMP #TA
  BEQ ADRTST
  CMP #TE
  BEQ ENTTST
  CMP #TB
  BNE *+5
  JMP BSTTST+B
  CMP #TI
  BNE *+5
  JMP INDEX+B
  CMP #TG
  BEQ GOTST
  BNE DATTST
;
ADRTST
  LDA #0
  STA FGENT
  JSR STRICH+B
  LDA #$7D;A
  STA DISP5
  LDA #$1F;D
  STA DISP6
  JSR ASK4+B
  JMP ENT1+B
;
ENTTST
  LDA DAT8
  LDX #0
  STA (ADR16,X)
  LDX #ADR16
  JSR INC16X+B
ENT1
  LDX #0
  LDA (ADR16,X)
  STA DAT8
  JMP TASTEN+B
;
GOTST
  LDA ADR16
  STA R1
  LDA ADR16+1
  STA R2
  LDA #0;0 FUER ANWE
  STA ADR16
  STA ADR16+1
  STA DAT8
  JSR STRICH+B
  LDA GA
  LDX GX
  LDY GY
  JMP (R1)
;
BSTTST
  LDX #ADR16
  JSR DEC16X+B
  JMP ENT1+B
;
DATTST
  LDX #$FF
  STX FGENT
DAT1
  INX
  CMP TTAB+B,X
  BNE DAT1
TSTGEF
  TXA
  ASL DAT8
  ASL DAT8
  ASL DAT8
  ASL DAT8
  ORA DAT8
  STA DAT8
  JMP TASTEN+B

;
INDEX ;VERTEILER
  JSR STRICH+B
  LDA #$14;1
  STA DISP5
  LDA #$0D;N
  STA DISP6
  JSR KEYIN+B
  JSR DISPSY+B
  CMP #$41;F
  BNE *+5
  JMP UHR+B
  CMP #$21;D
  BNE *+5
  JMP UHRA+B
  CMP #$42;B
  BNE *+5
  JMP INDEXB+B
  CMP #$31;E
  BNE *+8
  JSR STRICH+B
  JMP $E000 ;IC5
  CMP #$11;C
  BNE *+5
  JMP INDEXC+B;SHIFT
  JMP MON65+B
;
KEYINE
  JSR KEYIN+B
  CMP #TE
  BNE KEYINE
  RTS
KEYIN
  LDA KEY
  BEQ KEYIN
KEYIN1
  LDA KEY
  PHA
  LDA #0
  STA KEY
  PLA
  RTS
;
;*******************
;*   IRQ MONITOR    *
;*******************

IRQMON
  PHA
  TXA
  PHA
  TYA
  PHA


  TSX
  LDA $104,X
  AND #$10
  BEQ IRQ3 ;NO BRK
  JMP IRQBRK+B


IRQ3
  BIT IFR
  BVS IRQ4 ;NO IRQ
  LDA IRQF ;IRQ65?
  BNE IRQ4
  JMP (IRQ65)

IRQ4
  LDA IRQMF ;IRQ+MON
  BNE IRQ1 ;NO IRQM
  JSR IRQ5+B
  JMP IRQ1+B ;RETURN

IRQ5
  JMP (IRQM)

IRQ1
;******
;*******************
;* KEYBOARD'DISPLAY*
;*******************
;
  LDA LATC1L
  CLI

  LDA FGHEX
  BNE NOHEX
  JSR DISP16+B
  JSR DISP8+B
NOHEX; NEUE SP.
  JSR REAPA+B
  CLC
  ADC #$10
  CMP #$70
  BCS RESS;GR/GL
  STA PORTA
  BCC SETB ;SET PB
;
RESS; SET SP.1
  AND #$90
  STA PORTA
;
SETB
  LDX FGPB
  BNE SCAN ;OUTPB

  AND #$70
  LSR A   ;=>
  LSR A
  LSR A
  LSR A
  TAX
  LDA DISP1-1,X
  STA PORTB
;
SCAN ;ABFRAGE TASTEN
  JSR REAPA+B
  LDX RK1 ;ABSPRUNG
  DEX
  BEQ ABFR1
  BPL ABFR2
;
ABFR0 ;STAENDIG.ABFR
  TAX
  AND #$0F
  BEQ INTE;KEINE TA
  STX RKX
  INC RK1
  BNE INTE

ABFR1 ;TASTE TESTEN
  TAX
  AND #$70
  STA RI1
  LDA RKX
  AND #$70
  CMP RI1
  BNE INTE;SP.FALSC
  TXA
  CMP RKX
  BNE ABFRR ;FEHLER
  STA KEY
  INC RK1
  JMP INTE+B

ABFR2 ;TASTE LOS?
  CMP RKX
  BEQ INTE;DERUECKT
  AND #$70
  STA RI1
  LDA RKX
  AND #$70
  CMP RI1
  BNE INTE;FALS.SPA

ABFRR ;ABFRAGE RESET
  LDA #0
  STA RK1

INTE
;
;******
; UHRWERK
  SED
  DEC UHRR ;1/400
  BNE UEN2
  DEC UHRR+1
  BPL UEN2
  LDA #<400
  STA UHRR
  LDA #>400
  STA UHRR+1
  LDY #0
  LDX #2
UHR1
  LDA STD,X
  CLC
  ADC #1
  CPX #0
  BNE UHR2
  CMP #$24
  BCS UHR3
  BCC UEN1

UHR2
  CMP #$60
  BCC UEN1
UHR3
  STY STD,X
  DEX
  BPL UHR1
  BMI UEN2
UEN1
  STA STD,X
UEN2
  CLD
;---

ENDINT
  PLA
  TAY
  PLA
  TAX
  PLA
  RTI
REAPA ;READ PORTA
  LDA #$7F
  AND PORTA
  EOR #$0F
  RTS
;##################
DISPEX
  PHA
  LDA #$FF
  BNE *+5
DISPSY
  PHA
  LDA #0
  STA FGHEX
  JSR INIT1+B
  PLA
  RTS
;
;*******************
;* ANZEIGE ADRESSE *
;*******************
DISP16
  LDA ADR16+1
  JSR SEGN+B
  STX DISP1
  STY DISP2
  LDA ADR16
  JSR SEGN+B
  STX DISP3
  STY DISP4
  RTS
;
;*******************
;*  ANZEIGE DATUM  *
;*******************
DISP8
  LDA DAT8
  JSR SEGM+B
  STX DISP5
  STY DISP6
  RTS
;
SEGM       ;ACCU
  TAY
  LSR A  ;=>
  LSR A
  LSR A
  LSR A
  TAX
  LDA DISTAB+B,X
  TAX
  TYA
  AND #$0F
  TAY
  LDA DISTAB+B,Y
  TAY      ;4-7 0-3
  RTS      ; X   Y

;*******************
;*  SUCHE HEXZAHL  *
;*  DER TASTATUR   *
;*******************
SUCHEX
  JSR KEYIN+B
  CMP #TB
  BEQ SUB
  LDX #0
```

6.1 Microcomputer MPS 65

```
SUC1
  CMP TTAB+B,X
  BEQ SUC2
  INX
  CPX #16
  BEQ SUCHEX
  BNE SUC1
SUC2
  TXA
  AND #$0F
  CLC
  RTS
SUB
  SEC
  RTS
;
STRICH
  JSR PHAXY+B
  JSR INIT3+B
  JSR DISPEX+B
  LDX #05
  LDA #$08;-
STR1
  STA DISP1,X
  DEX
  BPL STR1
  JSR PLAXY+B
  RTS
;
UHR
  JSR STRICH+B
  LDA #$37;U
  STA DISP5
  LDA #$2D;H
  STA DISP6
  JSR ASK4+B
  LDA ADR16+1
  STA STD
  LDA ADR16
  STA MIN
  LDA #0
  STA UHRR
  STA UHRR+1
  STA SEK
UHRA
  JSR DISPSY+B
UHR0
  LDA STD
  STA ADR16+1
  LDA MIN
  STA ADR16
  LDA SEK
  STA DAT8
  JSR KEYIN1+B
  CMP #TE
  BNE UHR0
  JMP WARMST+B
;
IRGBRK ;00=BREAK
  LDA BRKUF
  BNE *+5
  JMP (BRKU)

  JSR DISPSY+B
  TSX
  LDA $103,X
  STA DAT8
  LDA $101,X
  STA ADR16
  LDA $102,X
  STA ADR16+1
  JMP MON65E+B
;
ASK4;LIST/SCHR. 4ANZ.
  JSR SUCHEX+B
  BCS *-3
  ASL A
  ASL A
  ASL A
  ASL A
  STA R1
  LDA DISTAB+B,X
  STA DISP1
  LDA ADR16+1
  AND #$0F
  ORA R1
  STA ADR16+1 ;ANZ 1
AS1
  JSR SUCHEX+B
  BCC *+8
  LDA #8
  STA DISP1
  BNE ASK4
  STA R1
  LDA DISTAB+B,X
  STA DISP2
  LDA ADR16+1
  AND #$F0
  ORA R1
  STA ADR16+1 ;ANZ 2
ASK2
  JSR SUCHEX+B
  BCC *+8
  LDA #8
  STA DISP2
  BNE AS1
  ASL A
  ASL A
  ASL A
  ASL A
  STA R1
  LDA DISTAB+B,X
  STA DISP3
  LDA ADR16
  AND #$0F
  ORA R1
  STA ADR16 ;ANZ 3
  JSR SUCHEX+B
  BCC *+8
  LDA #8
  STA DISP3
  BNE ASK2
  STA R1
  LDA DISTAB+B,X
  STA DISP4
  LDA ADR16
  AND #$F0
  ORA R1
  STA ADR16 ;ANZ 4
  JSR DISPSY+B
  RTS
;
INC16X
  INC 0,X
  BNE *+4
  INC 1,X
  RTS

DEC16X
  PHA
  DEC 0,X
  LDA 0,X
  CMP #$FF
  BNE *+4
  DEC 1,X
  PLA
  RTS
;
INIT1 ;VIA
  LDX #$7F
  STX IER ;CLEAR IFR

  LDX #VTABE-VTABA-1
INIT11
  LDA VTABA+B,X
  STA V,X
  DEX
  BPL INIT11

INIT3
  LDX #0
  STX FGPB ;DISP.EIN
  CLD
  CLI
  RTS

ERRINT      ;ERROR 6
  JSR INIT2+B
  LDA #6
  : ??MP ERR1+B
;
;*******************
;*  INIT VEKTOREN  *
;*******************

INIT2
  LDX #VEKE-VEKA-1
INIT21
  LDA VEKA+B,X
  STA RESF,X
  STA IRQF,X
  STA NMIF,X
  STA IRQMF,X
  STA BRKUF,X
  DEX
  BPL INIT21
  RTS

INIT4
  LDA #0
  STA GA
  STA GX
  STA GY
  LDX #9
CLEA1
  STA KEY,X
  DEX
  BPL CLEA1

  TAX ;RAM <=0
CLEA2
  STA $200,X
  STA $300,X
  STA $400,X
  STA $500,X
  STA $600,X
  STA $700,X
  INX
  BNE CLEA2
  RTS

NMI
  JSR PHAXY+B
  LDA NMIF
  BNE *+5
  JMP (NMI65)
  JMP ($FFFC)
;*******************
;*  INIT.-TABELLEN *
;*******************

VEKA
.BYT $FF       ;FLAG
.WOR ERRINT+B ;ERROR
VEKE

TAUF
.BYT $7D,$37,$69,$0D

VTABA
.BYT 0,$10,$7F,$70
.BYT <ZEIT1,>ZEIT1
.BYT 0,0,0,0,0,$40
.BYT 0,0,$C0
VTABE

;*******************
;* TASTEN-DISP.CODE*
;*******************

TER ;'ERROR'
.BYT $6B,9,9,$F,9
SPACE

*=ANFMON+$7D4
DISTAB;ANZ.0-F
.BYT $77,$14,$5B,$5E
,$3C,$6E,$6F,$54,$7F
,$7E,$7D,$2F
.BYT $63,$1F,$6B,$69
TAB65;,MPS-65
.BYT $39,$79,$6E,$08
.BYT $6F,$6E
TTAB;0-F
.BYT $18,$28,$38,$48
,$14,$24,$34,$44,$12
,$22,$32,$42
.BYT $11,$21,$31,$41
;
;*******************
;-INTERRUPTVEKTOREN-
;*******************

*=ANFMON+$7FA
.WOR NMI+B,MON65+B
.WOR IRQMON+B

    END
```

de 65 C02 serie.

De hardware.

Er is tegenwoordig een C-mos versie van de 6502 microprocessor in de handel. Het meest voor de hand liggende voordeel is natuurlijk het geringere stroomverbruik. Van deze C-mos 6502 zijn versies beschikbaar voor vier verschillende klokfrequenties te weten 1, 2, 3 en 4 MHz maximaal. Het opgenomen vermogen bedraagt respectievelijk 20mW, 40mW, 60mW en 80mW, dat is zelfs bij de snelste versie een flinke verbetering tegenover de bipolaire 6502, die 575 mW dissipeerde. De C-mos versie kent een standby-mode, waarin hij slechts 10μW! dissipeert.
De in- en uitgangen van de C-mos 6502 blijven echter TTL-compatibel. Inherent aan C-mos is de grotere storingsongevoeligheid en de ruimere tolerantie op de voedingsspanning (5V ± 20% in plaats van 5V ± 10%).

Er zijn twee versies (65 C102 en C112) beschikbaar voor een multiprocessor omgeving. Deze hebben twee extra aansluitingen nl. BE (bus enable) en ML (memory lock). Het bus enable signaal zet de processor in tri-state (ontkoppeld van de bus) indien het laag gemaakt wordt. Hierdoor komt het geheugen vrij voor gebruik door andere processoren of bijvoorbeeld DMA (direct memory access).
De 6502 mag echter nooit toegang tot het memory worden ontzegd gedurende read-modify-write instructies. Tijdens de uitvoering van dergelijke instructies is de ML-uitgang laag. Door dit signaal is gebruiken in een "bus-controle-circuit", kan worden voorkomen, dat BE laag wordt als ML laag is.
De res (reset)-ingang van de C-mos versies is voorzien van een Schmitt-trigger, waardoor met behulp van een RC-netwerk er een eenvoudige "power-on: reset" is te realiseren.

De software.

De instructieset van de 65Cxx versie is uitgebreider dan die van zijn bipolaire broer. Er zijn een aantal nieuwe instructies en voor een aantal bestaande instructies is er een adresseervorm bijgekomen. Zie tabel 7.

1. De nieuwe adresseervormen

- Voor de instructies ADC, AND, CMP, EOR, LDA, ORA, SBC en STA is er nu een indirecte adressering via pagina nul mogelijk. D.w.z.: direct na de instructie in het programma staat een byte, dat een adres op pagina nul weergeeft, op dit adres en het daaropvolgende in pagina nul staat het adres van de operand.

Voorbeeld:

instructie	adres	code
LDA ($10)	0210	B2
	0211	10
NOP	0212	EA
	0010	12
	0011	02

Bij uitvoering van de LDA ($10) instructie kijkt de processor op adres 10 van pagina nul, daar vindt hij het lage adresbyte van de operand, vervolgens vindt hij het hoge adresbyte op adres 0011. De accu wordt nu geladen met de inhoud van het gevonden adres. Het gevonden adres is 0212, dus de accu-inhoud wordt EA.

6.1 Microcomputer MPS 65

– Voor de jump-instructie is er ook een nieuwe adresseervorm nl. de geïndexeerde indirecte sprong JMP (IND), X. In dit geval wordt de jump-instructie gevolgd door een indirect adres, waarop het sprongadres staat. Bij dit sprongadres wordt eerst nog de inhoud van het X-register opgeteld, waarna naar het adres wordt gesprongen.

Voorbeeld:

instructie	adres	code
LDX #$02	0200	A2
	0201	02
JMP (0300), X	0202	7C
	0203	00
	0204	03
	0300	--
	0301	--
	0302	12
	0303	04
	0412	—
	programma wordt van hieraf voortgezet.	

N.B.: in de C-mos versie is ook de fout verdwenen die in de bipolaire versie voorkwam bij de JMP (IND)-instructie. Als het lage adresbyte nu op locatie FF van een pagina staat, wordt het hoge adresbyte gelezen van adres 00 van de volgende pagina (bij de bipolaire 6502 werd dit gelezen van adres 00 van dezelfde pagina!).

2. De nieuwe instructies.

Een aantal nieuwe instructies geven de 65C02 meer mogelijkheden tot het testen van de status en het manipuleren van individuele bits.

BIT
Deze instructie is eigenlijk niet nieuw, maar heeft er een aantal adresseringsmogelijkheden bij. Behalve absoluut en zero-page kan de instructie nu ook met immediate, zero-page X-indexed en absolute X-indexed adressering worden gebruikt.

TRB
Test and reset memory bits with accu.
functie: $A \wedge M$ naar M; M7 naar N; M6 naar V.
(A = accu, M = memory, N = negative flag, V = overflow flag)
adressering: ABS, Zero-page.

TSB
Test and set memory bits with accu.
functie: $A \vee M$ naar M; M7 naar N; M6 naar V.
adressering: ABS, Zero-page.

RMB0, RMB1 . . . . RMB7
Reset memory bit (0, 1 . . . . . 7)
functie: maak memory bit 0.
adressering: alleen Zero-page.

SMBO, SMB1 . . . . . . SMB7
Set memory bit (0, 1 . . . . . 7)
functie: maak memory bit 1.
adressering: alleen Zero-page.

BBR0, BBR1 . . . . . BBR7
Branch on memory bit reset (0, 1 . . . 7)
functie: test bit in geheugenplaats en branch als dit bit 0 is.

adressering:	Zero-page voor gerefereerde geheugenplaats, relatief voor de sprong.
instructievorm:	instructie. laag adresbyte van pagina nul sprongoffset.

6.1 Microcomputer MPS 65

BBS0, BBS1 . . . . . . BBS7
Branch on memory bit set (0, 1 . . . . . 7)
functie: test bit in geheugenplaats en branch als dit bit nul is.
adressering: als BBR.
instructievorm: als BBR.

BRA
Branch always.
functie: branch instructie zonder condities.
Deze instructie is nuttig bij het realiseren van relatieve (is plaats-onafhankelijke) programma's.

PHX – push (bewaar) X-register op de stack.

PHY – push (bewaar) Y-register op de stack.

PLX – pull x-register van stack. (haal een waarde van de stack en plaats deze in het X-register)

PLY – pull Y-register van stack, (haal een waarde van de stack en plaats deze in het Y-register)

STZ – store zero on memory. functie: maak geheugenplaats 00. adressering: absoluut; Zero-page; Zero-page X-indexed; absoluut X-indexed.

DEA – decrement accu. (verlaag accumulator met 1)

INA – increment accu. (verhoog accumulator met 1)

In tabel 7a en 7b zijn alle instructies met de erbij behorende codes vermeld.

In tabel 8 kan worden teruggevonden hoeveel klokcycli de processor voor een bepaalde instructie nodig heeft, zodat tijdkritische programma's van te voren op hun tijdsduur kunnen worden bekeken.

Alle niet gedefinieerde instructies worden door de C-mos versie van de 6502 behandeld als NOP. Bij de bipolaire versie is het gedrag niet altijd voorspelbaar.

6.1 Microcomputer MPS 65

	IMM	ABS	Z	ABS, X	ABS, Y	Z, X	Z, Y	(IND), Y	(IND), Y	(Z) IND	NB
ADC	2	4	3	4	4	4		6	5	5	(1)(2)
AND	2	4	3	4	4	4		6	5	5	(1)
ASL		6	5	7		6					
BIT	2	4	3	4		4					(1)
CMP	2	4	3	4	4	4		6	5	5	
CPX	2	4	3								
CPY	2	4	3								
DEC		6	5	7		6					
EOR	2	4	3	4	4	4		6	5	5	(1)
INC		6	5	7		6					
JMP		3						6		5	
JSR		6									
LDA	2	4	3	4	4	4		6	5	5	(1)
LDX	2	4	3		4		4				(1)
LDY	2	4	3	4		4					(1)
LSR		6	5	7		6					
ORA	2	4	3	4	4	4		6	5	5	
RMBn			5								
ROL		6	5	7		6					
ROR		6	5	7		6					
SBC	2	4	3	4	4	4		6	5	5	(2)
SMBn			5								
STA		4	3	5	5	4		6	6	5	
STX		4	3				4				
STY		4	3			4					
STZ		4	3	5		4					
TRB		6	5								
TSB		6	5								

IMPLIED INSTR.	**BRANCH INSTR.**	**NOTA BENE**
BRK 7	BBRn 5	1 Bij indexering 1 klokpuls meer indien paginagrens wordt overschreden
PHA 3	BBSn 5	2 Bij decimaal rekenen 1 klokpuls meer.
PHP 3	BCC 2	
PHX 3	BCS 2	
PHY 3	BEQ 2	
PLA 4	BMI 2	
PLP 4	BNE 2	Bij alle branch instructies.
PLX 4	BPL 2	1 klokpuls bijtellen als daadwerkelijk een sprong plaatsvindt en nogmaals 1 klokpuls extra indien daarbij een paginagrens wordt overschreden.
PLY 4	BRA 2	
RTI 6	BVC 2	
RTS 6	BVS 2	
Alle andere impl. instr. vergen 2 klokpulsen.		

Tabel 4/6.1 -8 "Speed"tabel van de 6502/65C02

6.1 Microcomputer MPS 65

Memory referenced **instructies**

	IMM	ABS	Z	ABS, X	ABS, Y	Z, X	Z, Y	(IND), X	(IND), Y	(Z) IND
ADC	69	6D	65	7D	79	75		61	71	72
AND	29	2D	25	3D	39	35		21	31	32
ASL		0E	06	1E		16				
BIT	89	2C	24	3C		34				
CMP	C9	CD	C5	DD	D9	D5		C1	D1	D2
CPX	E0	EC	E4							
CPY	C0	CC	C4							
DEC		CE	C6	DE		D6				
EOR	49	4D	45	5D	59	55		41	51	52
INC		EE	E6	FE		F6				
JMP		4C						7C		6C
JSR		20								
LDA	A9	AD	A5	BD	B9	B5		A1	B1	B2
LDX	A2	AE	A6		BE		B6			
LDY	A0	AC	A4	BC		B4				
LSR		4E	46	5E		56				
ORA	09	ØD	05	1D	19	15		01	11	12
RMB0			07							
RMB1			17							
RMB2			27							
RMB3			37							
RMB4			47							
RMB5			57							
RMB6			67							
RMB7			77							
ROL		2E	26	3E		36				
ROR		6E	66	7E		76				
SBC	E9	ED	E5	FD	F9	F5		E1	F1	F2
SMBO			87							
SMB1			97							
SMB2			A7							
SMB3			B7							
SMB4			C7							
SMB5			D7							
SMB6			E7							
SMB7			F7							
STA		8D	85	9D	99	95		81	91	92
STX		8E	86				96			
STY		8C	84			94				
STZ		9C	64	9E		74				
TRB		1C	14							
TSB		0C	04							

Tabel 4/6.1 -7a: De instructies van de 65C02

6.1 Microcomputer MPS 65

Implied instructies	
ASL(A)	0A
BRK	00
CLC	18
CLD	D8
CLI	58
CLV	B8
DEA	3A
DEX	CA
DEY	88
INA	1A
INX	E8
INY	C8
LSR (A)	4A
NOP	EA
PHA	48
PHP	08
PHX	FA
PHY	5A
PLA	68
PLP	28
PLX	FA
PLY	7A
ROL(A)	2A
ROR(A)	6A
RTI	40
RTS	60
SEC	38
SED	F8
SEI	78
TAX	AA
TAY	A8
TSX	BA
TXA	8A
TXS	9A
TYA	98

Branching instructies		
Inst.	code	vlag status (F) memory page 0 bit status (M)
BBR0	0F	M0 = 0
BBR1	1F	M1 = 0
BBR2	2F	M2 = 0
BBR3	3F	M3 = 0
BBR4	4F	M4 = 0
BBR5	5F	M5 = 0
BBR6	6F	M6 = 0
BBR7	7F	M7 = 0
BBS0	8F	M0 = 1
BBS1	9F	M0 = 1
BBS2	AF	M2 = 1
BBS3	BF	M3 = 1
BBS4	CF	M4 = 1
BBS5	DF	M5 = 1
BBS6	EF	M6 = 1
BBS7	FF	M7 = 1
BCC	90	F - C = 0
BCS	B0	F - C = 1
BEQ	F0	F - Z = 1
BMI	30	F - Z = 0
BNE	D0	F - N = 1
BPL	10	F - N = 0
BRA	80	geen conditie
BVC	50	F - V = 0
BVS	70	F - V = 1

Zie ook processor status register
BBR en BBS instructies zijn pagina nul instructies:
instr.; pag 0-adres; offset.

Tabel 4/6.1 -7b: De instructies van de 65C02 (vervolg)

6.1 Microcomputer MPS 65

00	BRK	1	40	RTI	1	80	BRA'	2	C0	CPY IMM	2
01	ORA (IND, X)	2	41	EOR (IND, X)	2	81	STA (IND, X)	2	C1	CMP (IND, X)	2
02	–	–	42	–	–	82	–	–	C2	–	–
03	–	–	43	–	–	83	–	–	C3	–	–
04	TSB Z	2	44	–	–	84	STY Z	2	C4	CPY Z	2
05	ORA Z	2	45	EOR Z	2	85	STA Z	2	C5	CMP Z	2
06	ASL Z	2	46	LSR Z	2	86	STX Z	2	C6	DEC Z	2
07	RMB0 Z	2	47	RMB4 Z	2	87	SMB0 Z	2	C7	SMB4 Z	2
08	PHP	1	48	PHA	1	88	DEY	1	C8	INY	1
09	ORA IMM	2	49	EOR IMM	2	89	BIT IMM	2	C9	CMP IMM	2
0A	ASL (A)	1	4A	LSR (A)	1	8A	TXA	1	CA	DEX	1
0B	–	–	4B	–	–	8B	–	–	CB	–	–
0C	TSB ABS	3	4C	JMP ABS	3	8C	STY ABS	3	CC	CPY ABS	3
0D	ORA ABS	3	4D	EOR ABS	3	8D	STA ABS	3	CD	CMP ABS	3
0E	ASL ABS	3	4E	LSR ABS	3	8E	STX ABS	3	CE	DEC ABS	3
0F	BBR0 Z	3	4F	BBR4 Z	3	8F	BBS0 Z	3	CF	BBS4 Z	3
10	BPL	2	50	BVC	2	90	BCC	2	00	BNE	2
11	ORA (IND), Y	2	51	EOR (IND), Y	2	91	STA (IND), Y	2	D1	CMP (IND), Y	2
12	ORA (IND)	2	52	EOR (IND), Z	2	92	STA (IND)	2	D2	CMP (IND), Z	2
13	–	–	53	–	–	93	–	–	D3	–	–
14	TRB Z	2	54	–	–	94	STY Z,X	2	D4	–	–
15	ORA Z,X	2	55	EOR Z,X	2	95	STA Z,X	2	D5	CMP Z,X	2
16	ASL Z,X	2	56	LSR Z,X	2	96	STX Z,Y	2	D6	DEC Z,X	2
17	RMB1 Z	2	57	RMB5 Z	2	97	SMB1 Z	2	D7	SMB5 Z	2
18	CLC	1	58	CLI	1	98	TYA	1	D8	CLD	1
19	ORA ABS, Y	3	59	EOR ABS, Y	3	99	STA ABS, Y	3	D9	CMP ABS, Y	3
1A	INA	1	5A	PHY	1	9A	TXS	1	DA	PHX	1
1B	–	–	5B	–	–	9B	–	–	DB	–	–
1C	TRB ABS	3	5C	–	–	9C	STZ ABS	3	DC	–	–
1D	ORA ABS, X	3	5D	EOR ABS, X	3	9D	STA ABS, X	3	DD	CMP ABS, X	3
1E	ASL ABS, X	3	5E	LSR ABS, X	3	9E	STZ ABS, X	3	DE	DEC ABS, X	3
1F	BBR1 Z	3	5F	BBR5 Z	3	9F	BBS 1 Z	3	DF	BBS5 Z	3
20	JSR	3	60	RTS	1	A0	LDY IMM	2	E0	CPX IMM	2
21	AND (IND, X)	2	61	ADC (IND, X)	2	A1	LDA (IND, X)	2	E1	SBC (IND, X)	2
22	–	–	62	–	–	A2	LDX IMM	2	E2	–	–
23	–	–	63	–	–	A3	–	–	E3	–	–
24	BIT Z	2	64	STZ Z	2	A4	LDY Z	2	E4	CPX Z	2
25	AND Z	2	65	ADC Z	2	A5	LDA Z	2	E5	SBC Z	2
26	ROL Z	2	66	ROR Z	2	A6	LDX Z	2	E6	INC Z	2
27	RMB2 Z	2	67	RMB6 Z	2	A7	SMB2 Z	2	E7	SMB6 Z	2
28	PLP	1	68	PLA	1	A8	TAY	1	E8	INX	1
29	AND IMM	2	69	ADC IMM	2	A9	LDA IMM	2	E9	SBC IMM	2
2A	ROL A	1	6A	ROR A	1	AA	TAX	1	EA	NOP	1
2B	–	–	6B	–	–	AB	–	–	EB	–	–
2C	BIT ABS	3	6C	JMP IND	3	AC	LDY ABS	3	EC	CPX ABS	3
2D	AND ABS	3	6D	ADC ABS	3	AD	LDA ABS	3	ED	SBC ABS	3
2E	ROL ABS	3	6E	ROR ABS	3	AE	LDX ABS	3	EE	INC ABS	3
2F	BBR2 Z	3	6F	BBR6 Z	3	AF	BBS2 Z	3	EF	BBS6 Z	3
30	BMI	2	70	BVS	2	B0	BCS	2	F0	BEQ	2
31	AND (IND), Y	2	71	ADC (IND), Y	2	B1	LDA (IND), Y	2	F1	SBC (IND), Y	2
32	AND (IND) Z	2	72	ADC (IND) Z	2	B2	LDA (IND) Z	2	F2	SBC IND	2
33	–	–	73	–	–	B3	–	–	F3	–	–
34	BIT Z,X	2	74	STZ Z,X	2	B4	LDY Z,X	2	F4	–	–
35	AND Z,X	2	75	ADC Z,X	2	B5	LDA Z,X	2	F5	SBC Z,X	2
36	ROL Z,X	2	76	ROR Z,X	2	B6	LDX Z,Y	2	F6	INC Z,X	2
37	RMB3 Z	2	77	RMB7 Z	2	B7	SMB3 Z	2	F7	SMB7 Z	2
38	SEC	1	78	SEI	1	B8	CLV	1	F8	SED	1
39	AND ABS, Y	3	79	ADC ABS, Y	3	B9	LDA ABS, Y	3	F9	SBC ABS, Y	3
3A	DEA	1	7A	PLY	1	BA	TSX	3	FA	PLX	1
3B	–	–	7B	–	–	BB	–	–	FB	–	–
3C	BIT ABS, X	3	7C	JMP (IND, X)	3	BC	LDY ABS, X	3	FC	–	–
3D	AND ABS, X	3	7D	ADC ABS, X	3	BD	LDA ABS, X	3	FD	SBC ABS, X	3
3E	ROL ABS, X	3	7E	ROR ABS, X	3	BE	LDX ABS, Y	3	FE	INC ABS, X	3
3F	BBR3 Z	3	7F	BBR7 Z	3	BF	BBS3 Z	3	FF	BBS7 Z	3

Tabel 4/6.1 -9: Decodeertabel voor 6502/65C02 programma's. Achter elke code staat vermeld hoeveel bytes de instructie inneemt.

6.1 Microcomputer MPS 65

4/6.1.1 Het geassembleerde monitorprogramma van de MPS 65 microcomputer

EPROM 2716 of een slechts voor de helft gebruikte 2732

```
000- 0C 1B FD 0C 1F FD 0C 05
008- FE 0C 0C FE 0C 93 FB 0C
010- 96 FB 0C 02 FF 0C 90 FE
018- 0C ED FB 0C EA FB 0C 94
020- FB 0C 0B FF 0C 12 FF 0C
028- F8 FA 0C 3E FF 0C 26 FE
030- 0C 0A FE 26 28 03 29 02
038- 19 20 14 08 01 0C 05 12
040- 31 2F 71 79 78 34 E5 E0
048- 85 DC E5 E1 85 DD 20 26
050- FE E9 6B 85 E7 E9 0D 85
058- E8 20 90 FE E5 E0 85 DE
060- E5 E1 85 DF 38 E5 DE E5
068- DC E5 DF E5 DD F0 05 E9
070- 01 0C 09 FA 20 92 FA 20
078- 26 FE E2 03 FD FB FF 95
080- E3 8A 10 F8 20 13 FD 78
088- E2 00 8E 00 E0 8A 8E 01
090- 01 86 E1 E2 7B 86 80 E2
098- 23 8E 00 01 E2 0D 8E 07
0A0- 01 8E 00 E0 E9 01 20 94
0A8- FB EE 00 E0 E2 02 EE 01
0B0- 01 90 10 E0 04 F9 D7 00
0B8- 99 02 01 88 10 F7 20 F8
0C0- F8 E2 08 E0 11 F0 1A E4
0C8- E2 F9 80 00 9D 00 01 E6
0D0- E2 E8 E5 81 18 69 07 85
0D8- E2 90 E8 20 F8 F8 0C 83
0E0- F8 20 12 F9 E2 00 FD 00
0E8- 01 20 6C F9 E8 E0 14 90
0F0- F5 E5 81 D0 EC 0C ED FB
0F8- 20 93 FB E9 00 85 E2 EA
100- 95 81 E8 E0 1C 90 F9 E5
108- DC 85 83 E5 DD 85 82 E5
110- E1 D0 04 85 81 F0 77 E2
118- 00 E1 DC E6 81 95 84 E6
120- 81 E6 DC D0 02 E6 DD 78
128- E5 DE E5 DC E5 DF E5 DD
130- F0 04 E6 E1 90 04 E0 17
138- D0 D5 E9 0D 95 87 E6 81
140- E8 E8 E9 00 18 75 81 CA
148- 10 FA E6 81 95 85 20 96
150- FB 20 E9 0C 8D 0C E0 E9
158- 80 8D 0B E0 E9 00 8D 05
160- E0 E2 20 E9 16 20 6C F9
168- 8A D0 F8 60 86 EA E0 07
170- 84 E7 08 85 E8 E2 02 E9
178- D0 8D 06 E0 E9 00 8D 07
180- E0 20 DF F9 06 E8 F0 0A
188- E9 E0 8D 06 E0 E9 01 8D
190- 07 E0 20 DF F9 CA 10 FA
198- 88 10 DA 68 E6 EA 60 2C
1A0- 0D E0 10 FB ED 04 E0 60
1A8- 20 D2 FA 78 E2 00 86 E2
1B0- 86 E1 8E 00 E0 CA 86 81
1B8- 20 0F FA 20 6E FA C9 23
1C0- F0 06 C9 16 D0 F2 F0 F3
1C8- E2 00 20 6E FA DD 00 01
1D0- E8 E0 11 D0 F5 E2 01 ED
1D8- 00 01 D0 1D E0 04 88 30
```

6.1 Microcomputer MPS 65

```
1E0- 14 F9 D7 00 C9 70 F0 F6
1E8- E0 03 F9 01 01 D9 D7 00
1F0- D0 86 88 10 F5 E6 E1 E2
1F8- 07 E5 E1 F0 FB E4 E2 E0
200- 10 F0 F5 FD 00 01 D9 80
208- 00 E6 E2 E8 E5 81 F0 7C
210- 18 69 07 C5 E2 D0 E2 E5
218- 82 85 DD E5 83 85 DC E5
220- 81 85 E0 85 E7 E0 00 84
228- E2 F9 84 00 91 DC D1 DC
230- D0 15 18 65 E0 85 E0 C8
238- C6 E7 D0 ED 18 65 82 18
240- 65 83 D9 85 00 F0 F2 E9
248- 05 20 F8 FA 0C ED FB E9
250- FF 8D 08 E0 20 7F FA 06
258- E7 05 E7 85 E7 C9 16 D0
260- F3 E2 05 20 6E FA C9 16
268- D0 EA CA D0 F6 60 E0 08
270- 20 7F FA 06 E7 05 E7 85
278- E7 88 D0 F4 69 FF 60 20
280- D1 FA 90 FB 20 D1 FA 20
288- D1 FA 20 D1 FA 6A 69 80
290- 60 2C 00 E0 70 1B 2C 00
298- E0 10 FB 65 00 ED 09 E0
2A0- 08 ED 08 E0 08 E9 FF 8D
2A8- 09 E0 68 C9 C7 68 E9 FE
2B0- 60 2C 00 E0 70 FB 10 E5
2B8- 20 66 FE 69 0F E8 E2 05
2C0- F9 D4 FF D5 E3 FD CD FF
2C8- CA 10 F8 20 13 FD 20 0C
2D0- FE 60 20 66 FE E9 69 85
2D8- E7 E9 21 85 E8 20 D0 FE
2E0- E5 E1 20 FA FB 86 D8 85
2E8- D7 E5 E0 20 FA FB 86 DA
2F0- 85 D9 E9 20 85 DB 60 E2
2F8- E0 20 12 FF E2 00 E1 E0
300- E2 E0 60 0B FF E2 07 DD
308- 74 FB F0 0B CA 10 F8 E9
310- 04 60 F8 FA 0C 7E FC E5
318- E0 08 E5 E1 08 20 66 FE
320- E9 6F C5 E7 E9 09 C5 E8
328- 60 D0 FE 68 85 E9 68 C5
330- E8 E5 E0 78 E5 E8 C5 EA
338- E5 E1 E5 E9 70 11 D0 68
340- C6 EA E5 EA C9 80 F0 60
348- E2 00 81 E8 0C 1D FB C9
350- FF D0 19 C6 EA E5 EA 10
358- 13 E2 00 81 E8 E5 E8 C5
360- E0 E5 E9 C5 E1 0C 7E FC
368- E9 03 D0 02 E9 02 60 F8
370- FA 0C 1D FB D0 F0 F0 70
378- D0 10 10 70 E2 FF CA FD
380- 00 02 DD 01 02 E9 EA DD
388- 00 02 E4 E0 D0 F0 E6 E0
390- 0C 7E FC 18 D0 01 78 C5
398- FF 68 C5 FD 68 C5 FE F0
3A0- 0B E5 FF 08 CA 08 D8 08
3A8- E5 FF D0 05 68 E8 68 EA
3B0- 68 E6 FD D0 02 E6 FE 6C
3B8- FD 00 08 60 C8 FB EA 68
3C0- 6A 6A 6A 6A 60 C8 FB 60
3C8- 69 0F C9 0A F0 03 09 70
3D0- 60 69 76 60 60 D3 FB E0
3D8- 0B E2 FF CA D0 FD 88 D0
3E0- F8 78 E9 01 D0 F1 60 D6
3E8- FB 60 60 13 FD E2 FF DA
3F0- 60 1F FF E9 73 C5 D3 D0
3F8- 04 C5 D4 F0 6F C5 D3 C5
```

6.1 Microcomputer MPS 65

```
400-  D4  60  7E  FF  20  D1  FF  60
408-  05  FE  E2  05  FD  E4  FF  D5
410-  E3  CA  10  F8  E9  14  60  D4
418-  FB  60  0C  FE  CE  03  E0  D0
420-  0B  EE  04  E0  D0  06  60  66
428-  FE  0C  05  E0  E5  C0  D0  03
430-  6C  C1  00  20  0C  FE  E9  00
438-  85  E0  E9  02  85  E1  E2  00
440-  E1  E0  85  E2  20  1B  FD  C9
448-  18  F0  64  C9  68  F0  75  C9
450-  64  D0  03  0C  F7  FC  C9  62
458-  D0  03  0C  D9  FC  C9  11  F0
460-  7A  C9  12  D0  03  0C  06  FB
468-  C9  14  D0  13  0C  E8  F9  E9
470-  00  85  E6  60  66  FE  E9  7D
478-  C5  E7  E9  1F  C5  E8  60  D0
480-  FE  0C  CF  FC  E5  E2  E2  00
488-  81  E0  E2  E0  60  0B  FF  E2
490-  00  E1  E0  C5  E2  E5  E3  F0
498-  F6  D0  E9  E5  E0  C5  E8  E5
4A0-  E1  C5  E9  E9  00  C5  E0  C5
4A8-  E1  C5  E2  60  66  FE  E5  ED
4B0-  E6  EE  E4  EF  6C  E8  00  E2
4B8-  E0  60  12  FF  0C  CF  FC  E2
4C0-  FF  C6  E6  E8  DD  EA  FF  D0
4C8-  FA  CA  06  E2  06  E2  06  E2
4D0-  06  E2  05  E2  C5  E2  0C  04
4D8-  FC  60  66  FE  E9  14  C5  E7
4E0-  E9  0D  C5  E8  60  1B  FD  60
4E8-  0C  FE  C9  01  D0  03  0C  7C
4F0-  FE  C9  61  D0  03  0C  DA  FE
4F8-  C9  02  D0  03  0C  F7  FA  C9
500-  71  D0  06  60  66  FE  0C  00
508-  E0  C9  11  D0  03  0C  7C  FB
```

```
510-  0C  ED  FB  60  1B  FD  C9  68
518-  D0  F9  60  E5  E3  F0  FC  E5
520-  E3  08  E9  00  C5  E3  68  60
528-  08  CA  08  D8  08  FA  FD  04
530-  01  69  10  F0  03  0C  F3  FE
538-  6C  0D  E0  70  07  E5  C3  D0
540-  03  6C  C4  00  E5  C9  D0  09
548-  60  0E  FD  0C  11  FD  6C  CA
550-  00  ED  04  E0  18  E5  E5  D0
558-  06  60  1A  FE  60  6D  FE  60
560-  FD  FD  18  69  10  C9  70  F0
568-  05  CD  01  E0  D0  05  69  D0
570-  CD  01  E0  E6  E4  D0  0C  69
578-  70  0A  0A  0A  0A  EA  F5  E2
580-  CD  00  E0  60  FD  FD  E6  EC
588-  CA  F0  0D  10  64  EA  69  0F
590-  F0  73  C6  EB  E6  EC  D0  6D
598-  EA  69  70  C5  ED  E5  EB  69
5A0-  70  C5  ED  D0  60  CA  C5  EB
5A8-  D0  17  C5  E3  E6  EC  0C  C5
5B0-  FD  C5  EB  F0  10  69  70  C5
5B8-  ED  E5  EB  69  70  C5  ED  D0
5C0-  04  E9  00  C5  EC  F8  C6  D5
5C8-  D0  6C  C6  D6  10  68  E9  D0
5D0-  C5  D5  E9  01  C5  D6  E0  00
5D8-  E2  02  F5  E9  18  69  01  E0
5E0-  00  D0  06  C9  64  F0  06  D0
5E8-  0B  C9  60  D0  07  D4  E9  CA
5F0-  10  E8  70  02  D5  E9  D8  68
5F8-  E8  68  EA  68  00  E9  7F  6D
600-  01  E0  09  0F  60  60  D3  FB
608-  E9  FF  D0  05  60  D3  FB  E9
610-  00  C5  E5  60  1F  FF  60  D6
618-  FB  60  E5  E1  60  77  FE  C6
```

6.1 Microcomputer MPS 65

```
620- E3 C4 E4 E5 E0 60 77 FE
628- C6 E5 C4 E6 60 E5 E2 60
630- 77 FE C6 E7 C4 E8 60 E8
638- 0A 0A 0A 0A EA FD D4 FF
640- EA D8 69 0F E8 F9 D4 FF
648- E8 60 60 1B FD C9 64 F0
650- 13 E2 00 DD EA FF F0 07
658- E8 E0 10 F0 ED D0 F4 CA
660- 69 0F 18 60 78 60 60 D3
668- FB 60 6F FF 60 05 FE E2
670- 05 E9 08 D5 E3 CA 10 FB
678- 60 D6 FB 60 60 66 FE E9
680- 63 C5 E7 E9 63 C5 E8 60
688- D0 FE E5 E1 C5 E9 E5 E0
690- C5 EA E9 00 C5 D5 C5 D6
698- C5 EB 60 0C FE E5 E9 C5
6A0- E1 E5 EA C5 E0 E5 EB C5
6A8- E2 60 1F FD C9 68 D0 ED
6B0- 0C 6C FC E5 CC D0 03 6C
6B8- CD 00 60 0C FE FA FD 03
6C0- 01 C5 E2 FD 01 01 C5 E0
6C8- FD 02 01 C5 E1 0C EA FB
6D0- 60 0A FE F0 FB 0A 0A 0A
6D8- 0A C5 E8 FD D4 FF C5 E3
6E0- E5 E1 69 0F 05 E8 C5 E1
6E8- 60 0A FE D0 06 E9 08 C5
6F0- E3 D0 DD C5 E8 FD D4 FF
6F8- C5 E4 E5 E1 69 F0 05 E8
700- C5 E1 60 0A FE D0 06 E9
708- 08 C5 E4 D0 DB 0A 0A 0A
710- 0A C5 E8 FD D4 FF C5 E5
718- E5 E0 69 0F 05 E8 C5 E0
720- 60 0A FE D0 06 E9 08 C5
728- E5 D0 D7 C5 E8 FD D4 FF
730- C5 E6 E5 E0 69 F0 05 E8
738- C5 E0 60 0C FE 60 08 E9
740- F0 C5 E4 CD 03 E0 68 CD
748- 00 E0 60 F6 00 D0 02 F6
750- 01 60 08 D6 00 F5 00 C9
758- FF D0 02 D6 01 68 60 E2
760- 7F CE 0E E0 E2 0E FD FF
768- FF DD 00 E0 CA 10 F7 E2
770- 00 C6 E4 D8 18 60 60 7E
778- FF E9 06 0C 09 FA E2 02
780- FD F8 FF D5 C0 D5 C3 D5
788- C6 D5 C9 D5 CC CA 10 F0
790- 60 E9 00 C5 ED C5 EE C5
798- EF E2 09 D5 E3 CA 10 FB
7A0- EA DD 00 02 DD 00 03 E8
7A8- D0 F7 60 60 D3 FB E5 C6
7B0- D0 03 6C C7 00 6C FC FF
7B8- FF 76 FF 6F 0F 08 08 00
7C0- 10 7F 70 C3 09 00 00 00
7C8- 00 00 00 00 00 C0 6B 09
7D0- 09 0F 09 06 77 14 1B 1E
7D8- 7C 6E 6F 14 7F 7E 7D 6F
7E0- 63 1F 6B 69 79 79 6E 08
7E8- 6F 6E 18 68 78 08 14 64
7F0- 74 04 12 62 72 02 11 61
7F8- 71 01 EB FF ED FB 68 FD
```

4/6.1.2

Onderdelenlijst en print-ontwerpen

Weerstanden, 1/8 W
R1 – R5 = 2,7 kΩ
R7 – R13 = 2,7 kΩ
R6 = 33 kΩ
R14 – R20 = 22 kΩ

Condensatoren
C1 = 10 pF, keramisch
C2 = 22 nF, MKH
C3 = 1 μF, 35 V

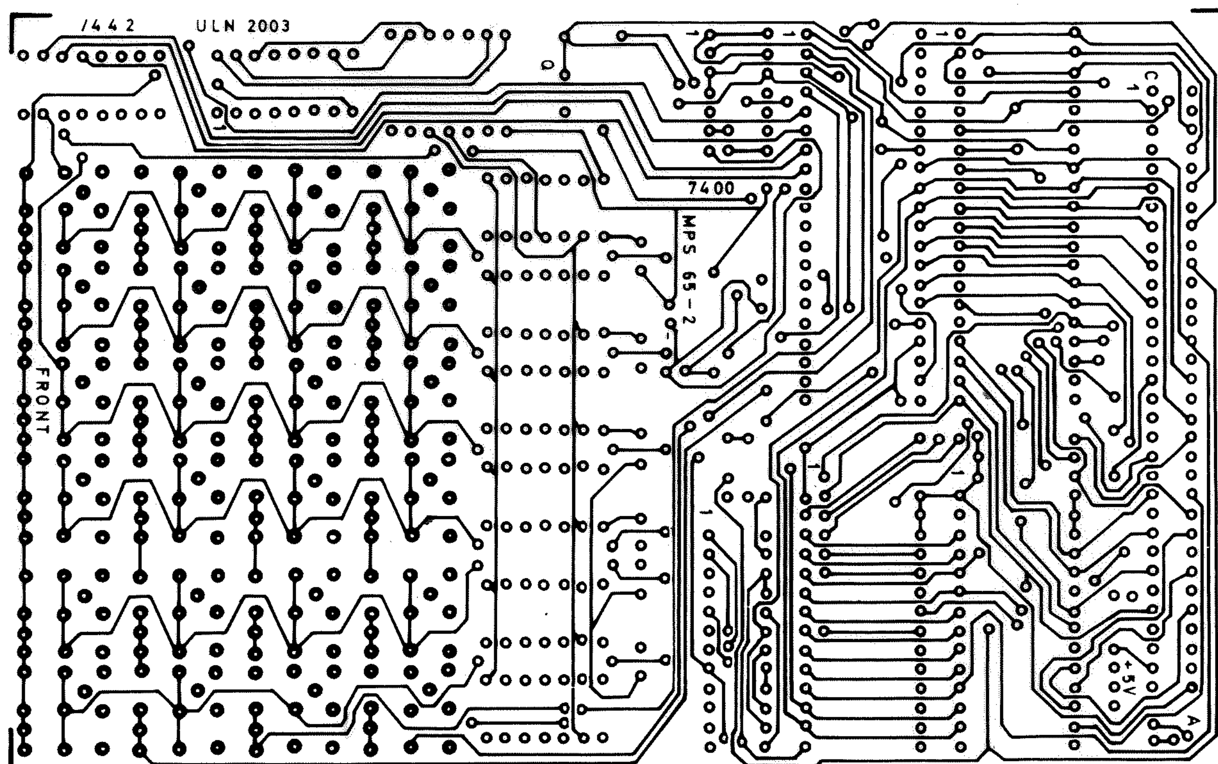
Halfgeleiders
T1 – T7 = BC 307 B
D1 – D6 = MAN 72 display's

Geïntegreerde schakelingen
IC1 = 6502 of 65C02 processor
IC2 = 6522 of 65C22 VIA
IC3 = 6116 RAM
IC4 = 2716 (2732) EPROM
IC5 = 7400 (74LS00) NAND-poort
IC6 = N 82 S 126 PROM
IC7 = ULN 2003 CMOS/TTL driver
IC8 = 7442 (74LS42) BCD/decimaal decoder

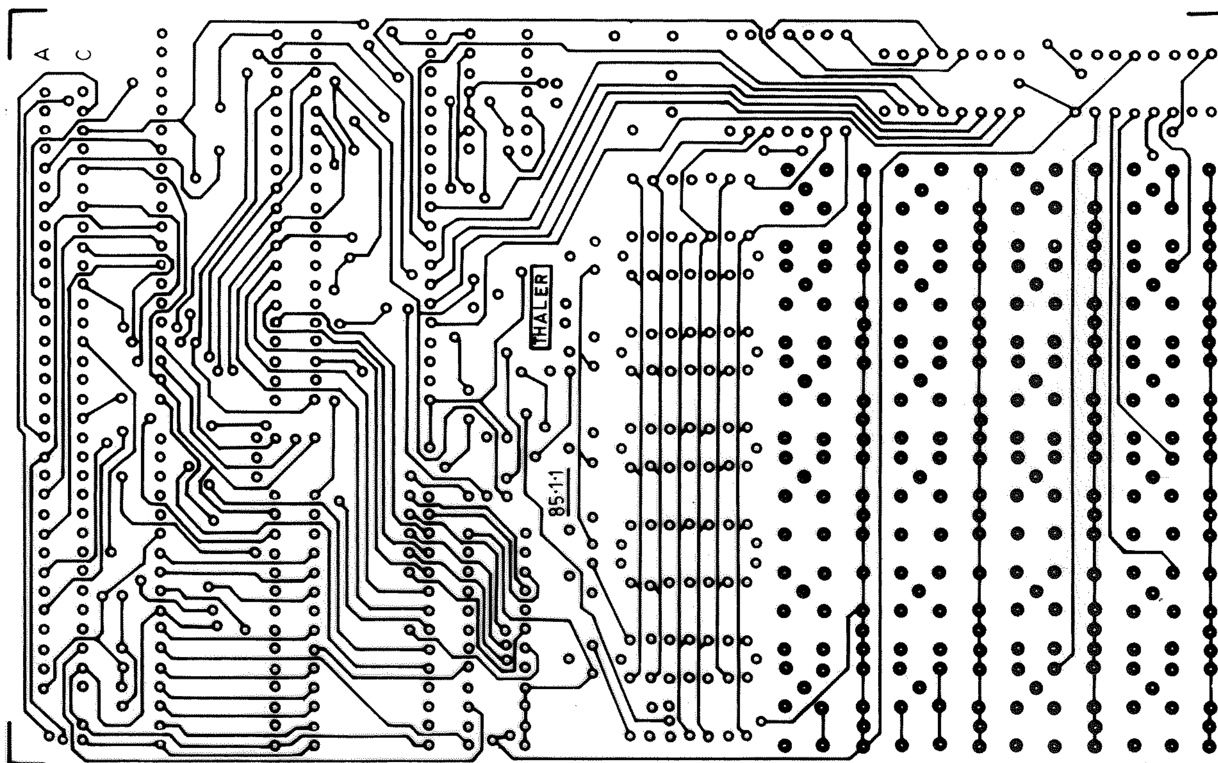
Diversen
Q = MEC 1.000 kristal
J1 = 64 polige connector volgens DIN 41612, type C
J2 = 4-polige connector, 2,54 mm raster
toetsen = 24 x digitast 17,1 x 12,3 mm

6.1 Microcomputer MPS 65

6.1 Microcomputer MPS 65



Figuur 4/6.1.2-1: Print-ontwerp, onderdelenzijde.



Figuur 4/6.1.2-2: Print-ontwerp, soldeerzijde.

4/6.2

Reset-schakelaar voor de C-64

Inleiding

Na het inschakelen van de voedingsspanning doorloopt de C-64 van Commodore, zoals iedere zich behoorlijk gedragende computer, een zogenaamde initialiserings- of reset-routine. Door deze routine worden alle elektronische schakelingen die dat nodig hebben in een bepaalde ondubbelzinnig bepaalde start-stand gezet.

Bij het testen van in machine-code geschreven programma's gebeurt het vaak dat de processor in een of andere lus blijft hangen, waardoor het apparaat niet meer reageert op het toetsenbord en het uit- en nadien weer inschakelen van de voeding de enige manier is om de elektronica weer onder menselijke controle te krijgen.

Nu is het voor geen enkele elektronische schakeling gezond als de voedingsspanning vaak kort na elkaar wordt uit- en weer ingeschakeld. Meestal treden er namelijk forse - hoewel zeer kortdurende - stroompulsen op die er op een kwade dag voor kunnen zorgen dat een van de ontelbare componenten waaruit zelfs een eenvoudige computer als de C-64 is samengesteld het welletjes vindt en de geest geeft.

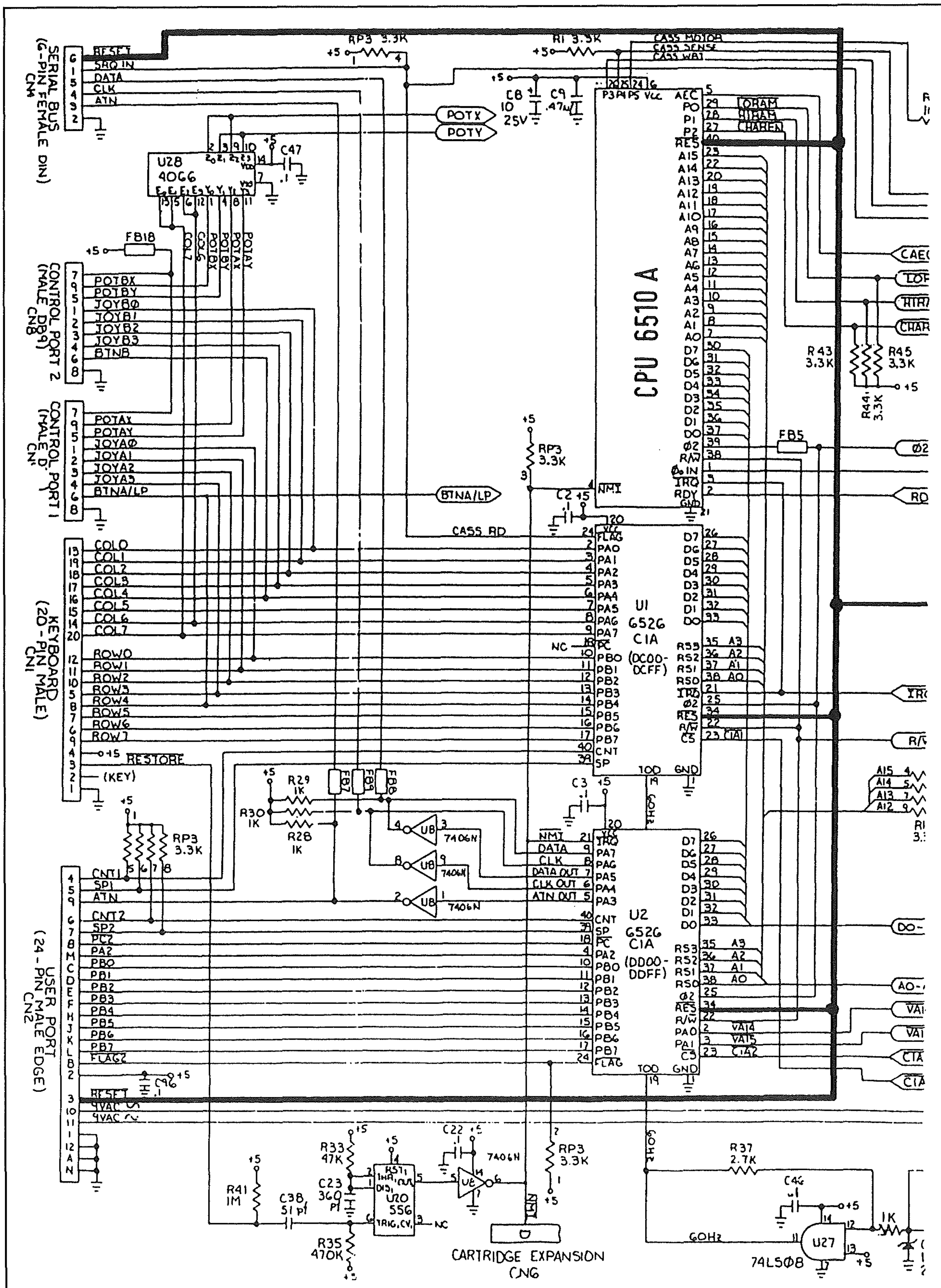
Een veel elegantere oplossing voor het uit een crash ontzetten van de elektronica bestaat in het aanbrengen van een zogenaamde reset-drukknop. Drukt men op die knop, dan wordt de initialisatieroutine doorlopen en de computer start op dezelfde manier op als had men hem voor korte tijd de voeding ontnomen.

Rechtstreeks op de processor

Iedere micro-processor bezit een RESET-ingang, die na het ontvangen van een geschikt signaal de processor naar een bepaald adres in het ROM-geheugen stuurt waar de initialisatie-routine begint.
Bij de Commodore 64 wordt een 6510 gebruikt als processor. De RESET zit, zoals uit het deelschema van figuur 4/6.2-1 blijkt, op pen 40 van het IC. Het streepje boven het woordje RES duidt er op dat deze ingang laag aktief is. Een reset wordt bijgevolg opgeroepen door deze pen even naar de massa te trekken. Uit het schema blijkt dat de RESET-lijn (vet ingetekend) rechtstreeks verbonden is met een pen van de seriële bus en met een aansluiting van de gebruikers-poort. Men kan een van deze aansluitingen zonder problemen door middel van een drukknopje rechtstreeks met de massa van het apparaat verbinden.

Het inbouwen van een reset-schakelaar zou in principe het meest ideaal opgelost kunnen worden via de seriële poort. Het volstaat immers een gestandaardiseerde en in iedere elektronica-winkel te verkrij-

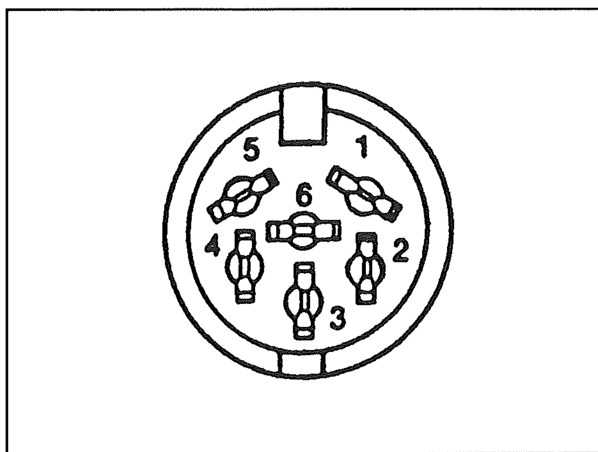
6.2 Reset-schakelaar voor de C-64



Figuur 4/6.2-1: Deel-schema van de ingewanden van de Commodore 64, waarin duidelijk de loop van het RESET-signaal van de processor kan worden gevolgd.

6.2 Reset-schakelaar voor de C-64

gen zespolige DIN-steker te kopen en de drukknop aan te sluiten tussen de pennen 6 (RESET) en 2 (MASSA). De aansluitcodering van zo'n zespolige steker is getekend in figuur 4/6.2-2.



Figuur 4/6.2-2: De zespolige DIN-steker voor de seriële poort van de Commodore 64.

Helaas heeft men bij het ontwerpen van de nieuwere versies van de C-64 om de een of andere reden besloten de RESET niet meer via de seriële poort naar buiten te brengen. Men moet dus eerst even in de appendix van de handleiding, waar de aansluitingen van alle connectoren en poorten staan beschreven, opzoeken of men dit truukje kan toepassen.

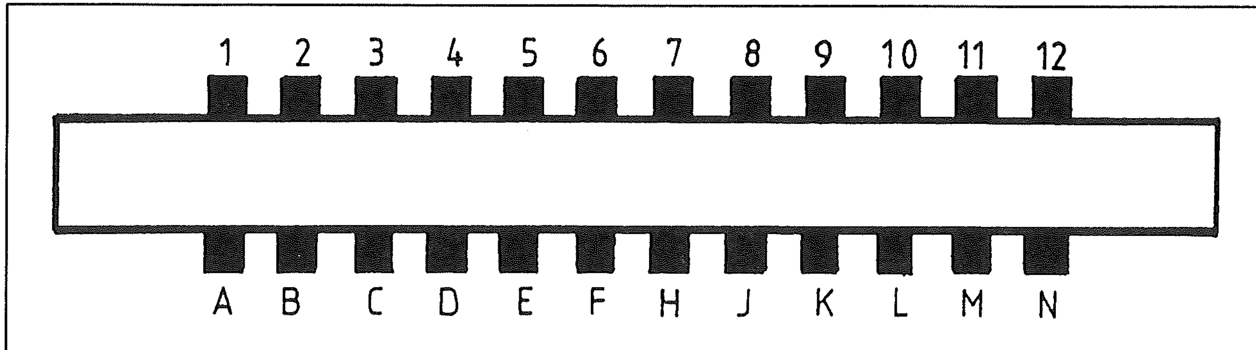
Gelukkig is de RESET-aansluiting op de gebruikers-poort wel behouden (dat zou ook niet anders kunnen!) en kan men op een iets ingewikkelder manier bij ieder model van deze populaire computer een RESET-schakelaar aanbrengen.

Via de gebruikers-poort
De gebruikers-poort (USER PORT) is uitgevoerd onder de vorm van een dubbelzijdige rand-connector van de print met twee maal 12 aansluitingen. De aansluitcode van deze connector volgt uit figuur 4/6.2-3, de aansluitcodes worden via tabel 4/6.2-1 gekoppeld aan de interne signalen van de C-64.

Denk er aan dat het niet mogelijk is de signalen van deze edge-connector af te takken door bijvoorbeeld klemmetjes op de koperen vlakjes te zetten! Deze maken contact met beide zijden van de print en men zal enige signalen met elkaar kortsluiten, iets dat de C-64 vast niet op prijs zal stellen! Men moet een speciale vrouwelijke connector voor dit doel aanschaffen, zie figuur 4/6.2-4. Dergelijke connectoren zijn redelijk goed verkrijgbaar en eventueel kan men een exemplaar per postorder onder code 74.06.91 bestellen bij bijvoorbeeld De Windmolen in Enschede. Handig is dat er speciale behuizingen voor dergelijke connectoren bestaan, zie figuur 4/6.2-5, waarin men de schakelaar kan monteren. Er ontstaat dan een handige en veilige reset-mogelijkheid die bij gebruik van andere perifere apparaten die van de user-port gebruik maken met een handbeweging uit de computer verwijderd kan worden. Ook die behuizing kan De Windmolen leveren, onder bestel-code 74.01.10.

De schakelaar wordt verbonden tussen de pennen 1 en 3 van de connector. De meest handige plaats voor de schakelaar is het gaatje in de behuizing waardoor men bij normaal gebruik van het onderdeel de kabel naar buiten voert. Of dit kan is volledig afhankelijk van de afmetingen van de enkelpolige drukschakelaar. Omdat er uiteraard geen stromen geschakeld worden en de spanning slechts 5 V bedraagt kan men de allerkleinste uitvoering toepassen die men kan verkrijgen.
Men kan de schakelaar met twee-compo-

6.2 Reset-schakelaar voor de C-64

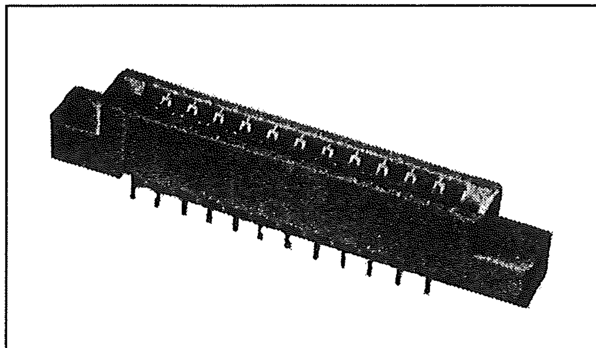


Figuur 4/6.2-3: De aansluitgegevens van de edge-connector voor de gebruikers-poort, die men aan de achterzijde van de computer aantreft.

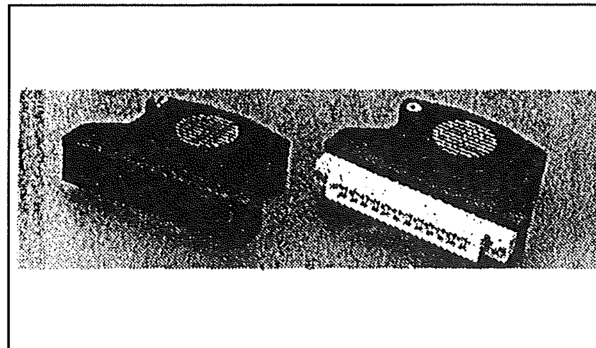
pen	signaal	opmerking
1	GND	
2	+5 V	max. 100 mA
3	$\overline{\text{Reset}}$	
4	CNT1	
5	SP 1	
6	CNT2	
7	SP 2	
8	$\overline{\text{PC 2}}$	
9	Ser. Atn In	
10	9 V AC	max. 100 mA
11	9 V AC	max. 100 mA
12	GND	
A	GND	
B	$\overline{\text{Flag 2}}$	
C	PB 0	
D	PB 1	
E	PB 2	
F	PB 3	
H	PB 4	
J	PB 5	
K	PB 6	
L	PB 7	
M	PA 2	
N	GND	

Tabel 4/6.2-1: De functie van de 24 pennen van de gebruikers-poort van de C-64.

6.2 Reset-schakelaar voor de C-64



Figuur 4/6.2-4: De 24-polige vrouwelijke connector, die over de edge-connector van de gebruikers-poort van de C-64 past.



Figuur 4/6.2-5: De behuizing voor de connector van figuur 4/6.2-4 is een handig hulpmiddel, omdat men de reset-schakelaar in het kabeldoorvoergaatje kan monteren.

nentenlijm in een van de helften van de behuizing van de connector lijmen. Nadien soldeert men de twee draadjes tussen de soldeerlipjes van de schakelaar en de genoemde aansluitingen van de connector. Tot slot lijmt men de tweede helft van de behuizing op de eerste en de schakelaar.

Belangrijke opmerking

Het nadeel van de gebruikers-poort van de Commodore is dat de vrouwelijke stekker op twee manieren in de edge-connector past! Het is dus absoluut noodzakelijk dat men de bovenzijde van de reset-combinatie duidelijk merkt. Zou men de reset-schakelaar per ongeluk 'op zijn kop' in de gebruikers-poort duwen en de drukknop bedienen, dan worden de aansluitingen N en L kortgesloten waardoor de PB-7 lijn met de massa wordt verbonden. Het is maar zeer de vraag of de 6526 CIA, die het data-verkeer tussen computer en gebruikers-poort regelt, deze mishandeling zou overleven!

6.2 Reset-schakelaar voor de C-64

4/6.4

Expansiepoort uitbreiding voor de C 64 met 24 in- en uitgangen

Inleiding

De voordelen van machinetaal
De Commodore C 64 is met in de miljoenen lopende verkoopcijfers een der meest populaire hobby computers, waarmee men talloze problemen met behulp van de programmeertaal BASIC kan oplossen. Het aantal mogelijkheden neemt echter drastisch toe, als men de computer in machinetaal programmeert. De kennis van machinetaal brengt vele voordelen met zich mee. Alleen met deze kennis kunnen de mogelijkheden van de computer volledig worden benut.

Het eerste voordeel is dat het aantal geheugenplaatsen nodig voor een machinetaal routine aanmerkelijk geringer is dan voor dezelfde routine in BASIC.
Een tweede voordeel is dat machinetaal programma's aanzienlijk sneller zijn dan BASIC programma's. Dit kan wel een faktor 100 schelen.
Een derde voordeel is dat veel bewerkingen, die via de userpoort of de expansiepoort lopen uitsluitend in machinetaal kunnen worden geprogrammeerd.

Een groot nadeel is echter dat machinetaal programma's niet zo eenvoudig direkt via het toetsenbord kunnen worden ingetikt. Dat gaat bij de meeste single board computers aanmerkelijk eenvoudiger. Daar kan men de hexadecimale code zo op een daartoe bestemd toetsenbordje ingeven. Op een display wordt dan meestal het geheugenadres, de instructiecode en de operand in HEX weergegeven. Aangezien de C 64 alleen direkt in BASIC is te programmeren kan het ingeven van een machinetaal programma alleen via een in BASIC geschreven hulpprogramma. Dit hulpprogramma dient als een machinetaal monitor, waarmee adressen en instructies kunnen worden ingegeven. Ook het bewaren, inlezen en starten van machinetaal programma's moet tot de funkties behoren.

Interfacen met de buitenwereld
Wat men met de computer zelf kan doen is beperkt tot wat op het scherm kan worden getoverd. Als men echter gebruik weet te maken van de aansluitingen naar de buitenwereld dan kan men de computer als een instrument gaan gebruiken. De interessantste verbinding met de buitenwereld is de expansiepoort. Via deze aansluiting is de CPU-bus van de C 64 naar buiten uitgevoerd. De op deze poort aangesloten IC's kunnen uitsluitend via machinetaal programma's worden aangestuurd.

Via de expansiepoort kunnen processen worden gestuurd waarbij een hoge snel-

heid van belang is. Met de funkties van op de expansiepoort aangesloten schakelingen kunt u meer indruk maken en plezier beleven dan met wat u op het beeldscherm weet te toveren.

Programmering in machinetaal
De processor werkt intern met 8-bit data. Een 8 bit datawoord of byte bestaat uit 8 informatie eenheden, die hoog of laag kunnen zijn (0 of 1). De processor werkt met slechts 2 niveau's. Aangezien het weergeven van getallen in een tweetallig systeem nogal onpraktisch is heeft men voor een hexadecimale weergave gekozen.

Het hexadecimale getalstelsel gebruikt de cijfers 0 tot en met 9 en de letters A tot en met F. Het is een kwestie van gebruiken om aan het hexadecimale stelsel te wennen. Soms is het nodig de status van individuele bits te weten. In dat geval moeten we toch teruggrijpen op het binaire stelsel. In de tabel van figuur 4/6.4-1 zijn de getallen van 0 tot 20 in de drie getalstelsels naast elkaar gezet.

Binair	Hexadecimaal	Decimaal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15
10000	10	16
10001	11	17
10010	12	18
10011	13	19
10100	14	20

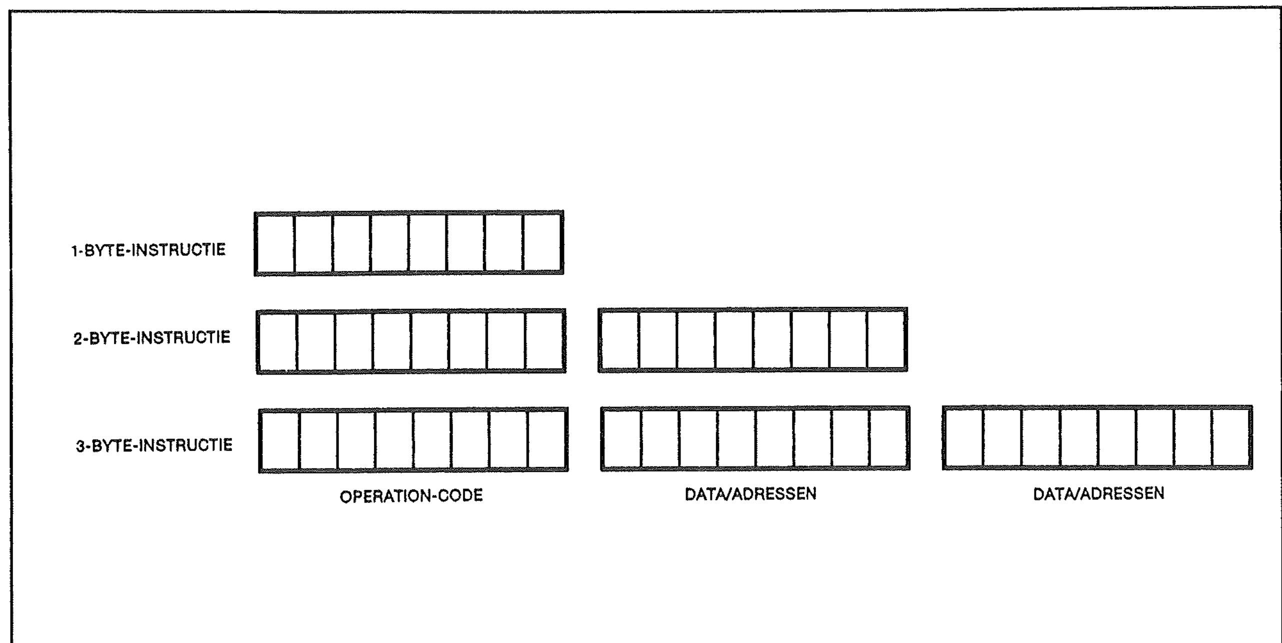
Figuur 4/6.4-1: De getallen 0 tot en met 20 in binaire, hexadecimale en decimale notatie.

De processor werkt met instructies die een, twee of drie bytes lang zijn, zie figuur 4/6.4-2.
In het eerste byte staat de code waarmee aangegeven wordt wat er gedaan moet worden, dus de eigenlijke instructie. In computertaal heet dit de operation code of kortweg opcode. In de andere bytes staan gegevens of adressen. Ieder byte wordt bij het programmeren in een bepaalde geheugenplaats in het lees/schrijf geheugen vastgelegd.
Het geheugenbereik voor machinetaal programma's is in de C 64 4 kB groot. Het begint op geheugenplaats C000 en eindigt op CFFF.

Vergis u niet in de omvang! Een machinetaal programma van 4 kB is groot te noemen. Indien noodzakelijk is het mogelijk een nog groter deel van het geheugen aan machinetaal programma's toe te wijzen.

Het programmeermodel van de C 64
De C 64 heeft vier 8-bit registers. Ook de stack is een 8-bit register. Het negende bit van de stack is nooit zichtbaar, maar altijd "1". De programmateller heeft 16-bit. Hiermee kunnen 65535 geheugenplaatsen worden geadresseerd. Directe data uitwisseling tussen adressen is niet mogelijk. Dit gaat steeds via een register.

6.4 Expansiepoort voor de C 64 met 24 in- en uitgangen



Figuur 4/6.4-2: Samenstelling van één, twee en drie bytes lange instructies.

Men kan de registers ook als tijdelijke opslagplaats zien, waarin echter bepaalde handelingen op de tijdelijk opgeslagen data kunnen worden uitgevoerd.

Een overzicht van dit programmeermodel is geschetst in figuur 4/6.4-3.

Byte, hex-code, instructieformaat
De kleinst in het geheugen te adresseren eenheid is een byte. De bits in een byte zijn genummerd van 0 tot en met 7.

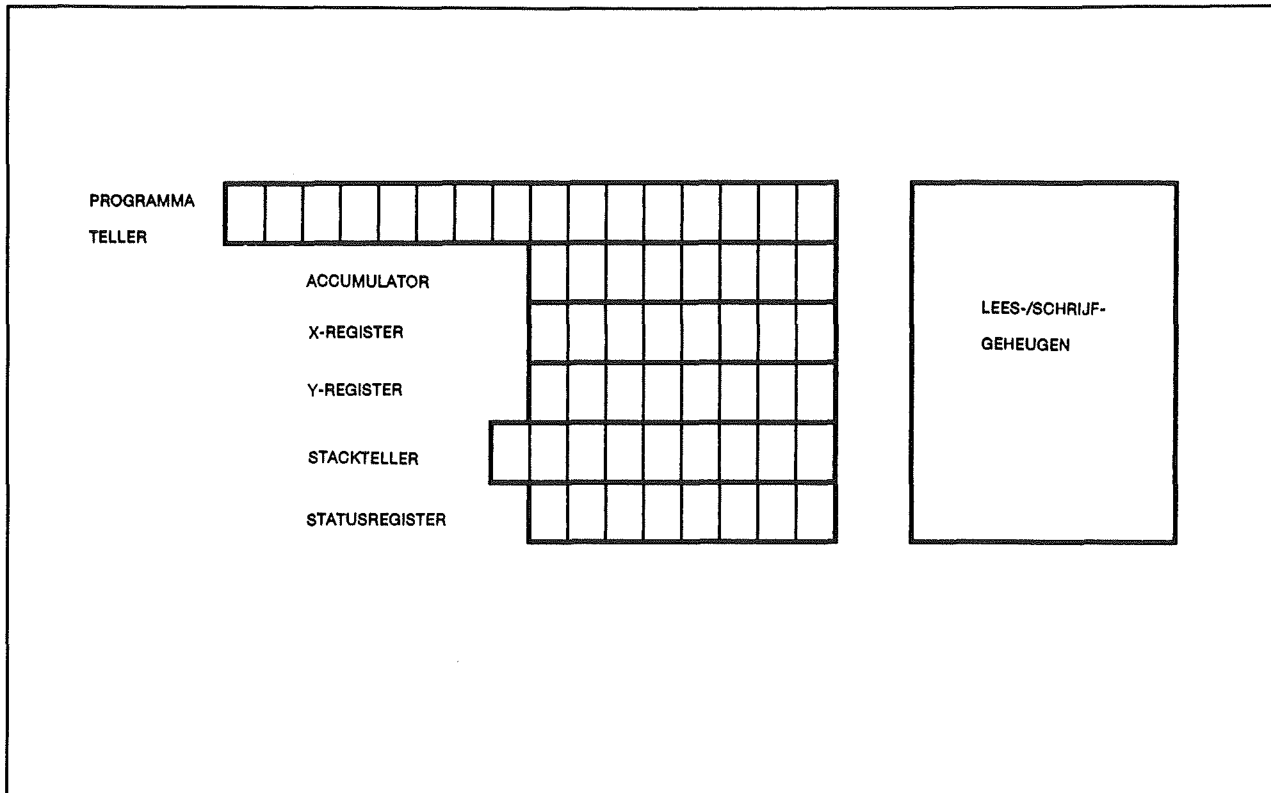
Wil men de decimale waarde van een byte weten, dan moet men de waarden van de plaatsen waar bits geset (1) zijn bij elkaar op tellen. Van rechts naar links heeft elke volgende positie de dubbele waarde van zijn voorganger. Het meest rechtse byte heeft de waarde 1. Hetzelfde geldt ook voor de adressen, waarbij elk adres echter uit 16 bit bestaat.

In figuur 4/6.4-4 is dit proces grafisch samengevat.

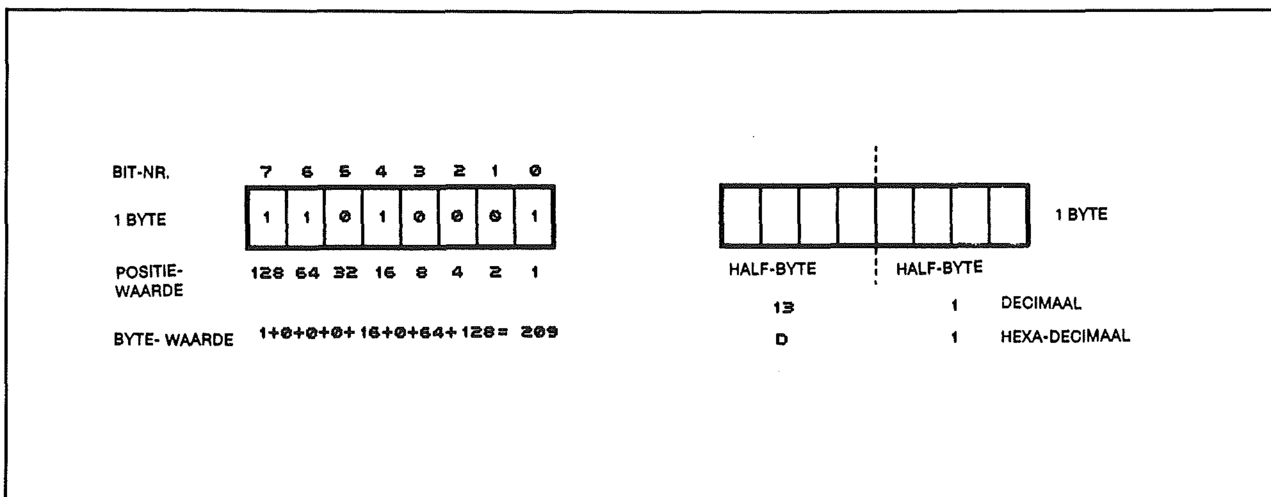
De instructieset
Uit de totale instructieset zijn een aantal instructies toegelicht die in de programma's in dit hoofdstuk worden gebruikt. Deze zijn samengevat in figuur 4/6.4-5 en in de tabellen die aan het einde van dit hoofdstuk zijn gepubliceerd. De instructieset van de in de C 64 gebruikte processor (een 6510) komt overeen met de 6502 van Rockwell. Voor beginners is het aan te raden eerst te proberen een aantal basisinstructies onder de knie te krijgen en daarmee wat programmeerervaring op te doen. Pas daarna kunt u zich veilig begeven op het gebied van de geïndexeerde adressering, waarmee op een elegante manier tabellen kunnen worden bewerkt.

De machinetaal monitor
De machinetaal monitor is een BASIC programma, dat met de "LOAD" opdracht in het geheugen wordt ingelezen. De volledige listing van dit programma is gegeven in listings 4/6.4-1, 4/6.4-2 en 4/6.4-3.

6.4 Expansiepoort voor de C 64 met 24 in- en uitgangen



Figuur 4/6.4-3: Het programmeermodel van de C 64.



Figuur 4/6.4-4: Voor de duidelijkheid wordt een byte uitgedrukt in twee hexadecimale digits. Voor een twee-byte woord zijn dat uiteraard vier hexadecimale digits noodzakelijk.

Na het laden kan met de toetsen STOP/RESTORE en de instructie "RUN" het programma worden gestart, waarna het menu op het scherm verschijnt. Met de toets "A" kunt u een beginadres opgeven.

6.4 Expansiepoort voor de C 64 met 24 in- en uitgangen

Instruktie	Mnemonic	Verklaring
A9XX	LDA (load accu)	Onmiddellijke adressering, de accu wordt geladen met de waarde XX, bijvoorbeeld met 3F. 2-bytes instruktie.
ADXXXX	LDA	Absolute adressering, de accu wordt geladen met de inhoud van geheugenplaats XXXX. Bijvoorbeeld met de inhoud van geheugenplaats C07F. 3-bytes instruktie.
8DXXXX	STA (store accu)	Absolute adressering, de inhoud van de accu wordt opgeslagen in geheugenplaats XXXX. Bijvoorbeeld in geheugenplaats C100. 3-bytes instruktie.
20XXXX	JSR (jump to subroutine)	Sprong naar subroutine op adres XXXX. 3-bytes instruktie.
60	RTS (return from subroutine)	Terugkeer van subroutine naar het hoofdprogramma. 1-byte instruktie.
A2XX	LDX	De waarde XX wordt geladen in het X-register. 1-byte instruktie.
A0XX	LDY	De waarde XX wordt geladen in het Y-register. 1-byte instruktie.
88	DEY	Verlaag de inhoud van het Y-register met 1. 1-byte instruktie.
CA	DEX	Verlaag de inhoud van het X-register met 1. 1-byte instruktie.
4CXXXX	JMP (jump)	Spring naar adres XXXX. 3-bytes instruktie.

Figuur 4/6.4-5: Enige instructies uit de instructieset van de C 64 die in dit hoofdstuk worden toegepast.

De computer verwacht een hexadecimaal getal van 4 cijfers.
Het adresbereik voor machinetaal programma's is C000 tot en met CFFF. Tikt u C100 in, dan verschijnt dit adres en de inhoud ervan op het scherm. Op dit moment kunt u reeds een nieuw programma invoeren.

6.4 Expansiepoort voor de C 64 met 24 in- en uitgangen

```
100 PRINT"♥"
110 PRINT"    MACHINETAAL MONITOR"
120 PRINT"    -------------------"
130 PRINT"  IN- EN UITVOER IN HEX"
140 PRINT
142 PRINT
150 PRINT"WERKEN MET HET PROGRAMMA (A)"
152 PRINT
160 PRINT"START VAN HET PROGRAMMA  (G)"
162 PRINT
170 PRINT"SCHRIJVEN OP CASSETTE    (S)"
172 PRINT
180 PRINT"LEZEN VAN CASSETTE       (L)"
190 PRINT
200 PRINT"----------------------------"
250 GET G$: IF G$="" GOTO 250
260 IF G$="A" THEN 1000
270 IF G$="G" THEN 2000
280 IF G$="S" THEN 3000
290 IF G$="L" THEN 4000
300 GOTO 250
1000 PRINT"♥"
1010 PRINT"   ADRES INVOER"
1020 PRINT
1030 PRINT"VOLGEND ADRES SPATIE, VORIG ^"
1032 PRINT"    MET RETURN NAAR MENU"
1040 PRINT
1050 INPUT"BEGINADRES";H4$
1060 GOSUB 5000
1070 S=D4+1
1080 PRINT
1090 S=S-1
1095 GOSUB 5100
1100 PRINT"   ";
1110 S0=PEEK(S)
1120 N=3:GOSUB 5150
1130 PRINT" - ";
1140 D2=0
1150 FOR I=2 TO 1 STEP-1
1160 GET G$:IF G$="" GOTO 1160
1170 IF G$="^" THEN 1080
1180 IF G$=" " THEN 1260
1190 T=ASC(G$)
1200 IF T=13 GOTO 100
1210 PRINT G$;
1220 F=48: IF T>58 THEN F=55
1230 D2=D2+(T-F)*I*I*I*I
1240 NEXT I
1250 POKE S,D2
1260 S=S+1
1270 PRINT
1280 GOTO 1095
```

Listing 4/6.4-1: De machinetaal monitor, deel 1.

6.4 Expansiepoort voor de C 64 met 24 in- en uitgangen

```
2000 PRINT"♥"
2010 PRINT"  START HET MACHINETAAL PROGRAMMA"
2020 PRINT
2030 INPUT"STARTADRES";H4$
2040 GOSUB 5000
2050 SYS(D4)
2060 GOTO 200
3000 PRINT"♥"
3005 PRINT"    SCHRIJVEN NAAR CASSETTE"
3010 PRINT
3020 INPUT"STARTADRES";H4$
3030 GOSUB 5000
3040 L=D4
3050 PRINT
3060 INPUT"EINDADRES";H4$
3070 GOSUB 5000
3080 H=D4
3090 D=H-L
3100 PRINT
3110 FS=-1:FD=-1:FC=-191
3120 INPUT"PROGRAMMA";F$
3130 OPEN1,1,1,F$
3140 J=LEN(STR$(D))
3150 FS=FS+J
3160 PRINT#1,D
3170 FOR J=0 TO D
3180 I=PEEK(L=J)
3190 PRINT#1,I
3200 K=LEN(STR$(I))
3210 FE=INT(-(FS+K)/FC)
3220 IF FE=FD GOTO 3260
3230 T=TI
3240 IF (TI-T)>5 GOTO 3240
3250 FS=-1
3260 FS=FS+K
3270 NEXT J
3280 CLOSE 1
3290 PRINT
3300 PRINT"DATA VASTGELEGD OP CASSETTE"
3310 GOTO 200
4000 PRINT"♥"
4010 PRINT"   LEZEN VAN CASSETTE"
4020 PRINT
4030 INPUT"PROGRAMMANAAM";F$
4040 PRINT
4050 OPEN 1,1,0,F$
4060 INPUT #1,D
4070 INPUT"STARTADRES";H4$
4080 GOSUB 5000
4090 PRINT
4100 L=D4
4110 FOR I=0 TO D
```

Listing 4/6.4-2: De machinetaal monitor, deel 2.

6.4 Expansiepoort voor de C 64 met 24 in- en uitgangen

```
4120 INPUT #1,X
4140 POKE L+I,X
4150 NEXT I
4160 CLOSE 1
4165 PRINT:PRINT
4170 PRINT"PROGRAMMA INGELEZEN"
4180 GOTO 200
4230 PRINT" ";
5000 REM OMZETTING HEX NAAR DEC.
5010 D4=0: M=4096
5020 FOR I=1 TO 4
5030 T=ASC(MID$(H4$,I,1))
5040 F=48: IF T>58 THEN F=55
5050 D4=D4+(T-F)*M
5060 M=M/16
5070 NEXT I
5080 RETURN
5100 REM GEEFT 2 OF 4 HEX DIGITS
5110 N=1
5120 S(1)=INT(S/4096)
5130 S(2)=INT((S-S(1)*4096)/256)
5140 S0=(S-S(1)*4096)-S(2)*256
5150 S(3)=INT(S0/16)
5160 S(4)=(S0-S(3)*16)
5170 FOR K=N TO 4
5180 F=48:IF S(K)>9 THEN F=55
5190 PRINT CHR$(S(K)+F);
5200 NEXT K
5210 RETURN
READY.
```

Listing 4/6.4-3: De machinetaal monitor, deel 3.

Een voorbeeldje wordt gegeven in listing 4/6.4-4.

Het programma wordt gestart met "G"C100. De aanhalingstekens worden niet getikt, deze geven slechts aan, dat het om de letter G gaat. Het programma is onmiddellijk klaar. Geven we nu het adres van C050 in ("A"C050), dan zult u zien dat hierin de waarde 55 is opgeslagen. Let erop, dat bij het ingeven van het adres in een instructie eerst het laagste byte van het adres wordt ingegeven en dan pas het hoogste. Dus in adres C102 niet ingeven 8DC050 maar zoals aangegeven 8D50C0.

Het bewaren en laden van machinetaal programma's volgt uit de aanwijzingen op het scherm, namelijk met opgave van de programmanaam, beginadres en eindadres.

Vertragingen

Zelfs de naar huidige maatstaven niet zo snelle C 64 computer voert bewerkingen nog in een naar menselijke begrippen razendsnel tempo uit.

Vaak is het gewenst dat een programma op een of meerdere plaatsen een korte pauze inlast. Dit kunnen we bereiken, door in een subroutine een bepaalde instructie een van te voren bepaald aantal malen te laten uitvoeren. De instructie heeft geen nuttig effekt, maar de uitvoering ervan kost wel tijd en die tijd is onze gewenste pauze.

6.4 Expansiepoort voor de C 64 met 24 in- en uitgangen

```
C100    A955      LDA   Laad de accu met de waarde 55
C102    8D50C0    STA   Bewaar de accu-inhoud in adres C050
C105    00        BRK   (Break) Einde programma.
```

Listing 4/6.4-4: Een klein voorbeeldje van het gebruik van de machinetaal monitor.

In het programma in listing 4/6.4-5 wordt het X-register geladen met het getal FF (255 decimaal). Met de instructie CA in regel CF12 wordt de inhoud van het X-register met 1 verminderd.
De nieuwe waarde zal dus FE (254 decimaal) zijn. In regel CF13 wordt onderzocht of de inhoud van het X-register al nul is. Dat is uiteraard op dit moment nog niet het geval. Er wordt nu teruggesprongen naar CF12. Hier wordt het X-register weer met 1 verminderd. FE min 1 is FD, met als gevolg dat in regel CF13 weer wordt teruggesprongen. Pas na 255 maal zal de inhoud van het X-register nul zijn.

De voorwaarde om terug te springen is dan niet meer aanwezig en de computer zal de instructie van regel CF15 uitvoeren. Hier staat 60 (RTS), waarmee de computer uit de vertragingssubroutine terugkeert naar het hoofdprogramma. Voor de hier getoonde routine heeft de C 64 circa 1 milliseconde nodig. Kortere tijden kunnen worden gerealiseerd door in plaats van FF in regel CF11 een kleiner getal te nemen, bijvoorbeeld 05.

Met het programma van listing 4/6.4-6 kunnen al langere vertragingen worden gerealiseerd. Hierin wordt het Y-register geladen met FF en in een lus telkens met 1 verminderd. Met de inhoud van het X-register wordt bepaald hoe vaak de inhoud van het Y-register van FF tot 00 moet worden teruggeteld. Het programma doorloopt circa 65000 maal de programmaregels CF04/CF05.
Met deze routine kan een vertraging tot 0,3 sekonde worden gerealiseerd.

Programmavertakkingen
In nagenoeg alle programma's komt een aantal vertakkingen voor. Vanaf zo'n positie kan het programma hetzij gewoon worden vervolgd, hetzij op grond van een bepaalde conditie op een ander adres worden voortgezet. Bij vertakkingen wordt niet aangegeven naar welk adres moet worden gesprongen, maar wordt een positieve of negatieve verplaatsing ten opzichte van het huidige adres aangegeven. Het is van belang te weten hoe deze verplaatsing wordt berekend.

Veel gebruikte vertakkingsinstructies zijn:

- BNE (D0), Branch if not equal to 0
  Spring aangegeven aantal plaatsen voor of achteruit als nulvlag niet gelijk is aan 0.
- BEQ (F0), Branch if equal to 0
  Spring aangegeven aantal plaatsen voor of achteruit als nulvlag gelijk is aan 0.

6.4 Expansiepoort voor de C 64 met 24 in- en uitgangen

```
CF10   A2FF   LDX       Laadt het X-register met FF

CF12   CA     DEX       Verminder de inhoud van het
                        X-register met 1. (Als de
                        inhoud van het X-register 0
                        wordt, wordt de nulvlag
                        geset).
CF13   DOFD   BNE       Als de nulvlag niet is geset
                        spring dan 3 adressen terug.
                        (FD = -3 ! ). De programma-
                        teller staat al op CF15.
                        Dus spring naar CF12
CF15   60     RTS       Keer terug naar het hoofd-
                        programma.
```

Listing 4/6.4-5: Een voorbeeldprogramma dat een korte vertraging introduceert.

Als voorbeeld wordt een kort programma besproken.

Stel dat op adres C120 de vertakkingsinstructie BEQ (F0XX) staat. De bedoeling is dat wanneer aan de gestelde conditie wordt voldaan (nulvlag geset) de computer het programma voortzet op adres C138. De computer haalt de vertakkingsinstructie BEQ = F0 XX uit het geheugen. De programmateller staat dan op adres C122. Om XX te bepalen gaat het er dus om het verschil uit te rekenen tussen het doeladres C138 en de huidige stand van de programmateller C122.
In hexadecimaal is C138 - C122 = 16. Men kan nu in de instructie voor XX dus 16 invullen. Op adres C120 zet men dus F016. Wordt bij het uitvoeren van de vertakkingsinstructie aan de voorwaarde voldaan, dan wordt de programmateller met 16 verhoogd en de volgende instructie zal van adres C138 worden gehaald. Dit is een sprong vooruit. Uiteraard kan het ook gewenst zijn een sprong achteruit te maken. Ook hier is het zaak het verschil tussen de huidige programmatellerstand en het adres van de instructie waar naar toe gesprongen moet worden vast te stellen. De programmateller zal op het adres van de instructie staan die volgt op de vertakkingsinstructie. Stel dat de vertakkingsinstructie BNE op adres CF13 staat. De instructie zelf omvat 2 bytes, dus de programmateller staat op CF15. Vlak voor de vertakkingsinstructie staat een DEX instructie waar naar terug gesprongen wordt, zoals in de tijdvertragingsroutines. De DEX instructie staat op adres CF12. Dit is het doeladres. Volgens dezelfde berekeningsmethode als bij de voorwaartssprong, levert CF12 - CF15 een negatief getal op namelijk -3. Nu is per definitie vastgelegd, dat de computer alle getallen waarvan bit 7 op 1 staat als negatief intepreteert. Dus positieve getallen zijn 00 tot en met 7F, negatief zijn 80 tot en met FF, waarbij FF = -1, FE = -2, etc. Dat wil zeggen, dat met vertakkingsinstructies niet verder kan worden gesprongen dan 127 (= 7F HEX) vooruit of -128 (= 80 HEX) achteruit.
Zonder al te diep in te gaan op de theorie kan men een sprong berekenen door het huidige programmatelleradres in HEX af te trekken van het doeladres. Denk eraan dat als er 1 geleend wordt van de volgende digit dit de waarde 16 heeft.

6.4 Expansiepoort voor de C 64 met 24 in- en uitgangen

```
CF00   A2FF   LDX        Laadt het X-register met FF

CF02   A0FF   LDY        Laadt het Y-register met FF

CF04   88     DEY        Verminder de inhoud van het
                         Y-register met 1. (Als de
                         inhoud van het Y-register 0
                         wordt, wordt de nulvlag
                         geset).
CF05   D0FD   BNE        Als de nulvlag niet is geset
                         spring dan 3 adressen terug.
                         (FD = -3 ! ). Dus naar CF04
CF07   CA     DEX        Verminder de inhoud van het
                         X-register met 1. (Als de
                         inhoud van het X-register 0
                         wordt, wordt de nulvlag
                         geset).
CF08   D0F8   BNE        Als de nulvlag niet is geset
                         spring dan 8 adressen terug.
                         (F8 = -8 ! ). Dus naar CF02
CF0A   60     RTS        Keer terug naar het hoofd-
                         programma.
```

Listing 4/6.4-6: Een programma dat een langere vertragingstijd tot gevolg heeft.

Bovendien is F = 15, E = 14, ..., A = 10, zie het onderstaande voorbeeld.

```
C F 1 2     →C F 0 2+16     →C F 0 18
C F 1 5 -   →C F 1 5 -      →C F 1 5 -
-----------  --------------  -------------
                                       13

C F 0       →C E 0+16       →C E 16
C F 1 -     →C F 1 -        →C F 1 -
-----------  --------------  ------------
                                       15

C E      →B E+16 →B 14+16→B 30
C F -    →C F -   →C 15 -   →C 15 -
--------  ---------  ---------  -------
                                    15
```

Er wordt nu nog 1 van de niet bestaande vijfde digit geleend:

```
B        →B+16    →11+16    →27
C -      →C -     →12 -     →12 -
-------  -------  --------  -------
                               15
```

Als antwoord ontstaat in decimaal:
15 15 15 13
en in hexadecimaal:
FFFD

Als men uit de eerste twee digits een getal krijgt dat niet gelijk is aan FF probeert men een grotere sprong te maken dan is toegestaan. In dat geval moet een JMP (jump) instructie worden ingeschakeld.

De expansiepoort uitbreiding

De werking van de schakeling

Het principe-schema van de uitbreidingsschakeling is getekend in figuur 4/6.4-6, het volledige praktische schema in figuur 4/6.4-7.

De datalijnen D0 tot en met D7 van het poort-IC 8255 zijn direkt met de databus van de C 64 verbonden.
Voor de adres-selectie zorgen de drie lijnen I/01 - CS, A0 en A1.
Hieruit worden voor de 8255 de volgende adressen verkregen:

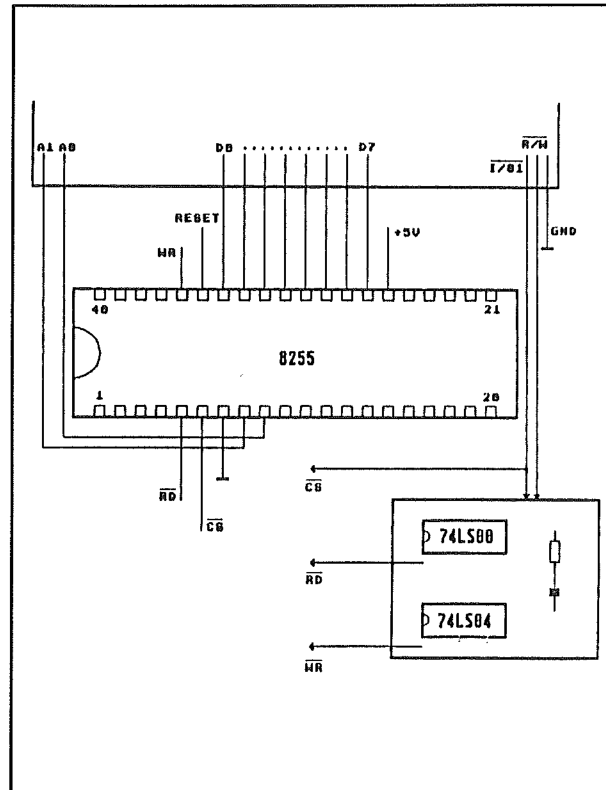
- poort A - DE00;
- poort B - DE01;
- poort C - DE02;
- controlpoort DE03.

De lees/schrijf informatie uit de C 64 voldoet niet zonder meer aan de eisen die de specificaties van de 8255 stellen. Met een aantal poorten, weerstanden en condensatoren worden de signalen van de C 64 aangepast aan de 8255. De resetlijn van de 8255 is verbonden met aarde.

De bouw van de schakeling

In de figuren 4/6.4-8 en 4/6.4-9 zijn respectievelijk de koper- en de componentenzijde van de dubbelzijdige print getekend.

In figuur 4/6.4-10 is de onderdelen plattegrond gegeven.
Let op de punten die met een kruisje (x) zijn aangegeven, deze moeten doorgecontacteerd worden. Dat wil zeggen, dat het eilandje aan de onderzijde van de print wordt verbonden met het eilandje aan de bovenzijde.



Figuur 4/6.4-6: Het principe-schema van de uitbreiding.

Het makkelijkst gaat dit door in het gat een klein stukje draad te steken, dit aan beide zijden te solderen en de eindjes af te knippen. Ook de punten B1-B1 en B2-B2 moeten met elkaar worden verbonden zoals aangegeven op de tekening.

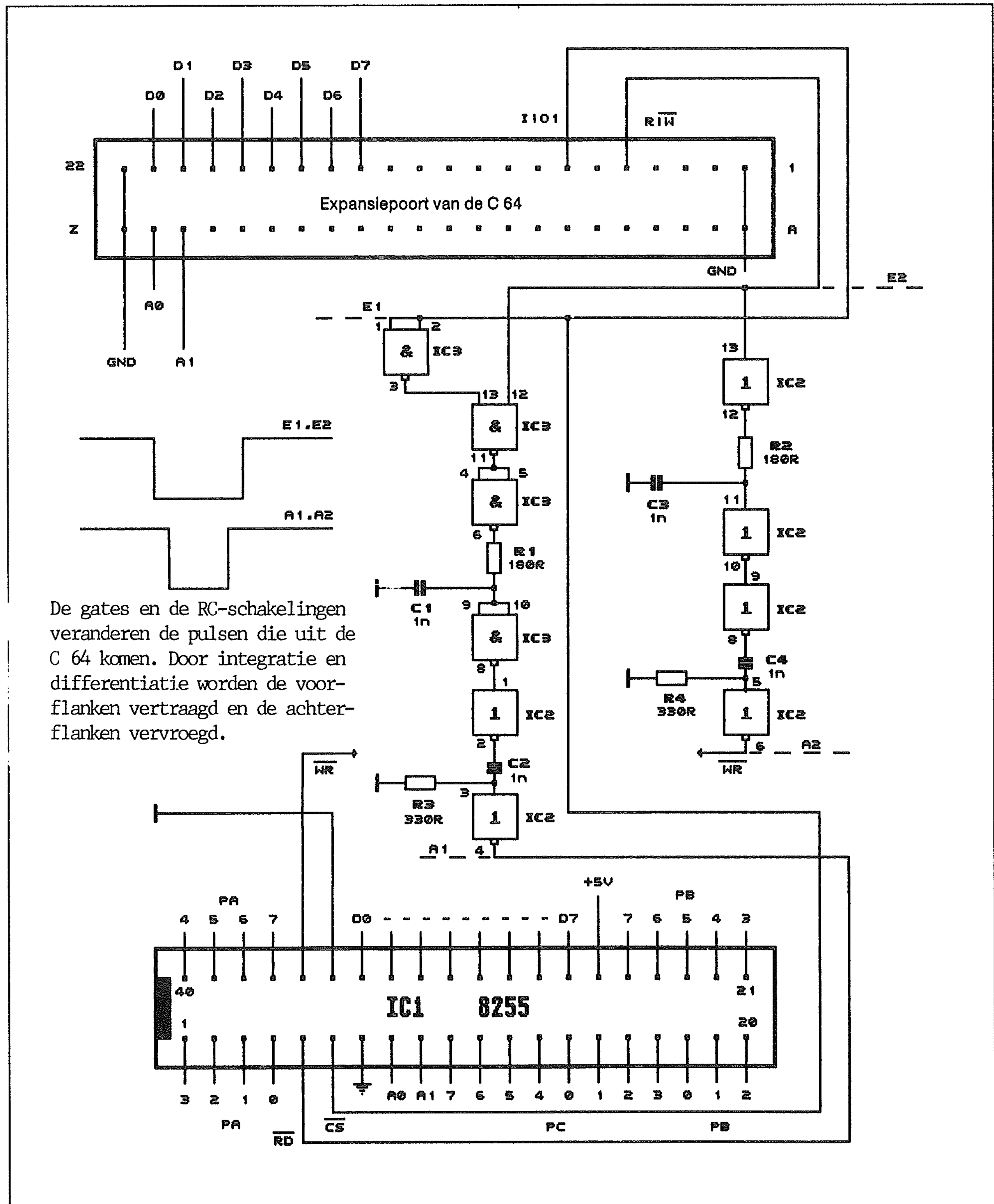
De schakeling heeft als voeding een spanning van +5 V nodig.

Mogelijkheden van de schakeling

De drie poorten van de 8255 hebben elk 8 in/uitgangen. Hiermee staan 24 lijnen ter beschikking, die naar believen als in- of uitgang kunnen worden geprogrammeerd. Dit wordt gedaan middels het control-adres (DE03).

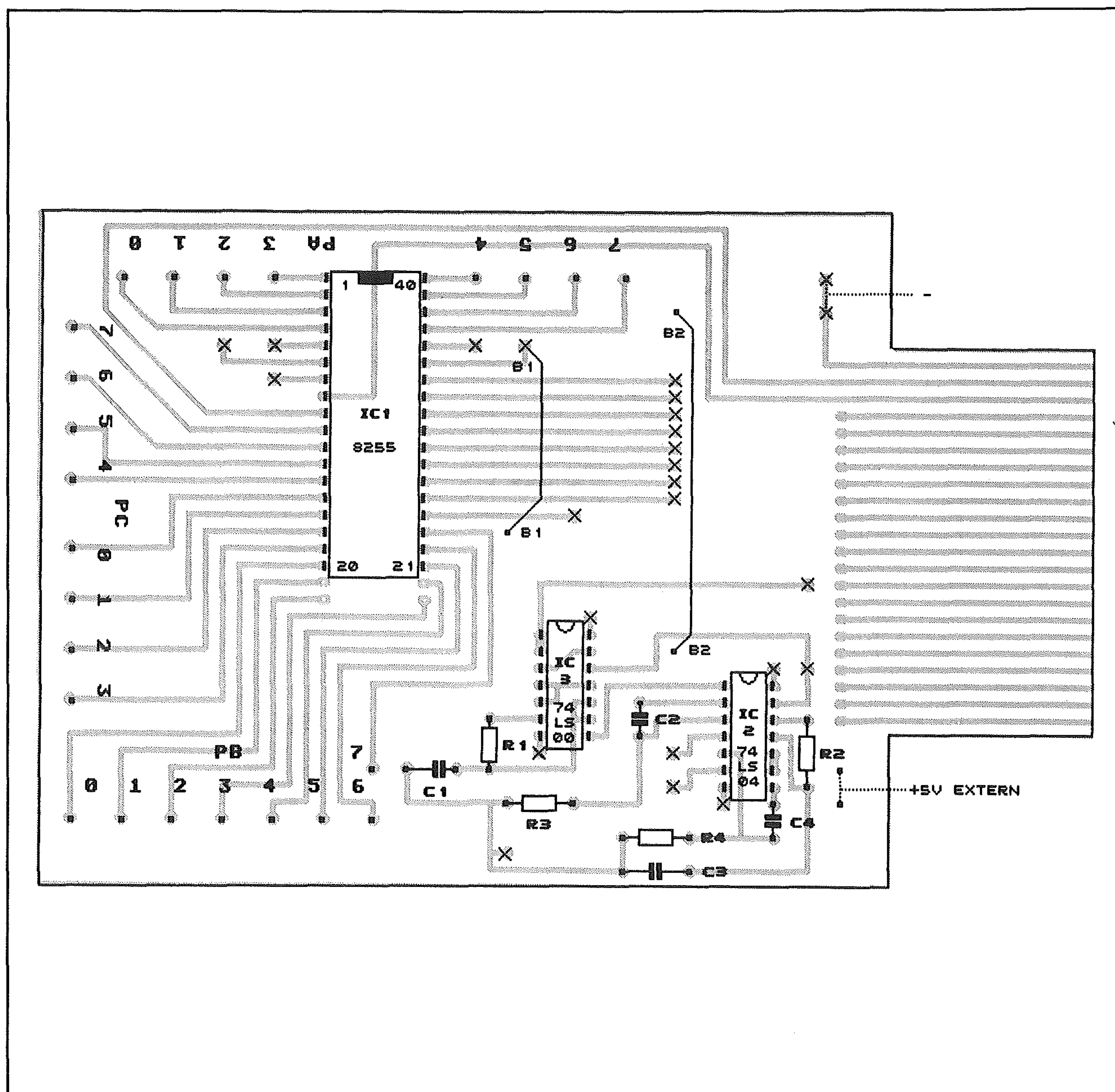
Van de talrijke combinatiemogelijkheden worden enige voorbeelden besproken.

6.4 Expansiepoort voor de C 64 met 24 in- en uitgangen



Figuur 4/6.4-7: Het praktische schema van de uitbreidingschakeling.

6.4 Expansiepoort voor de C 64 met 24 in- en uitgangen



Figuur 4/6.4-10: Componentenopstelling van de print.

- Voorbeeld 1
  PA0-PA7 = 8 x uitgang
  PB0-PB7 = 8 x uitgang
  PC0-PC7 = 8 x uitgang
  Hiermee heeft men de beschikking over 24 uitgangen.
  Deze instelling wordt bereikt door in het control-adres 80 te schrijven (STA).

- Voorbeeld 2
  PA0-PA7 = 8 x uitgang
  PB0-PB7 = 8 x uitgang
  PC0-PC3 = 4 x ingang
  PC4-PC7 = 4 x uitgang
  Hiermee heeft men de beschikking over 20 uitgangen en 4 ingangen. In het control-adres moet hiertoe 81 worden geschreven.

6.4 Expansiepoort voor de C 64 met 24 in- en uitgangen

- Voorbeeld 3
  PA0-PA7 = 8 x uitganG
  PA0-PB7 = 8 x ingang
  PC0-PC3 = 4 x ingang
  PC4-PC7 = 4 x uitgang
  Hiermee heeft men de beschikking over 12 ingangen en 12 uitgangen. In het control-adres wordt 83 geschreven.

- Voorbeeld 4
  PA0-PA7 = 8 x uitgang
  PB0-PB7 = 8 x ingang
  PC0-PC7 = 8 x ingang
  Hierbij beschikt men over 16 ingangen en 8 uitgangen. Het bijbehorende controlewoord is 89.

De controle logika wordt gestuurd door te schrijven in adres DE03. Om de gewenste combinatie volgens voorbeeld 2 te programmeren moet men in het machinetaal programma de volgende instructies opnemen:
- A9 81 (LDA)
  laadt accu met controlewoord 81
- 8D 03 DE (STA)
  bewaar de inhoud van de accu in DE03

De in- en uitgangen van de 8255 kunnen ook als adressen worden aangesproken.

Een voorbeeld:
- A9 0F (LDA)
  laadt accu met bit patroon 00001111
- 8D 00 DE (STA)
  bewaar de inhoud van de accu in DE00

Na uitvoering van deze instructies staan op poort A spanningen, die overeenkomen met het geprogrammeerde bitpatroon 00001111.

Om de status van de aansluitingen van poort C in de computer binnen te halen moet een leesaktie worden ondernomen. Uiteraard moeten de lijnen van poort C eerder als ingangen zijn geprogrammeerd. In machinetaal ziet het lezen van poort C er als volgt uit:

- AD 02 DE (LDA)
  laadt de accu met de inhoud van geheugenadres DE02

DE02 is poort C van de 8255.

Het programma kan nu verder gaan met bijvoorbeeld compare of logische instructies (AND, OR, XOR) om vast te stellen welke bits van de poort hoog zijn en welke laag.

Onderdelenlijst

Weerstanden, 1/4 W:
R1,R2 = 180 Ω
R3,R4 = 300 Ω

Condensatoren:
C1,C2,C3,C4 = 1 nF

Geïntegreerde schakelingen:
IC1 = SAS 8255A
(poortschakeling van AMD, Intel, NSC, NEC, Oki, Toshiba)
IC2 = SN 74 LS 04
IC3 = SN 74 LS 04

Testprogramma's

Testprogramma en testschakeling 1
Met behulp van de schakeling van figuur 4/6.4-11 kan het expansiepoort IC worden getest. De uitgangen van de expansiepoort worden verbonden met drijvers. Deze sturen signaallampjes of LED's aan. Als men alle 24 poorten van zo'n drij-

6.4 Expansiepoort voor de C 64 met 24 in- en uitgangen

ver/lamp combinatie voorziet dan kan men in een oogopslag de toestand van elk bit van de drie poorten zichtbaar maken. Zijn 24 drijver/lamp schakelingen wat te veel van het goede, dan moet men toch minstens 8 lampjes aansluiten zodat men tenminste een poort in z'n geheel kan bekijken. Het programma van listing 4/6.4-7 is een uitvoerprogramma. Dat wil zeggen, dat alle 24 lijnen van de expansiepoort zijn gebruikt als uitgang. Het programma zet een aantal bit patronen op de uitgangen volgens figuur 4/6.4-12.

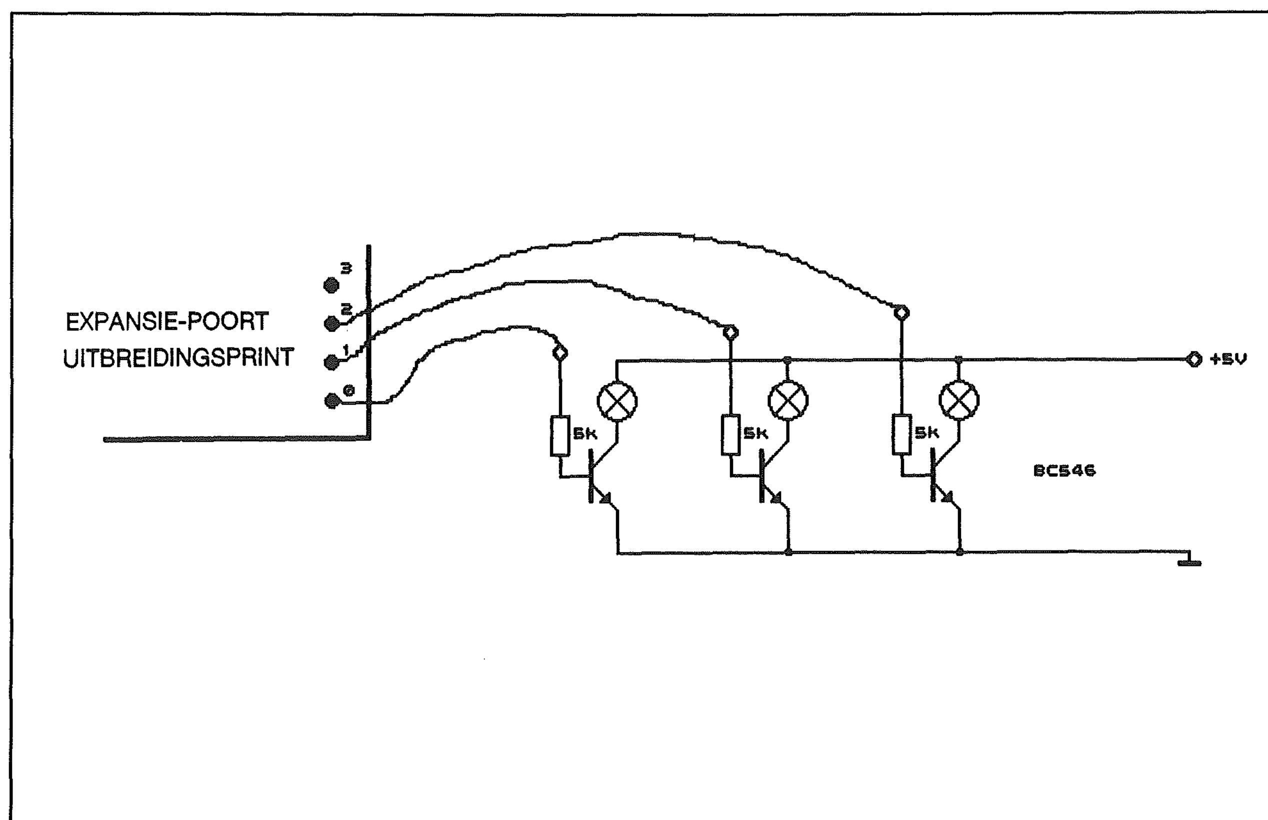
Testprogramma en testschakeling 2

Een eenvoudige schakeling waarmee de lijnen PC0 tot en met PC3 als ingangen worden gebruikt is getekend in figuur 4/6.4-13.

Bitvoorbeeld								Hexwaarde
0	0	0	0	1	1	1	1	0F
1	1	1	1	0	0	0	0	F0
1	1	1	1	1	1	1	1	FF
0	1	0	1	0	1	0	1	55
1	0	1	0	1	0	1	0	AA
0	0	0	0	0	0	0	0	00

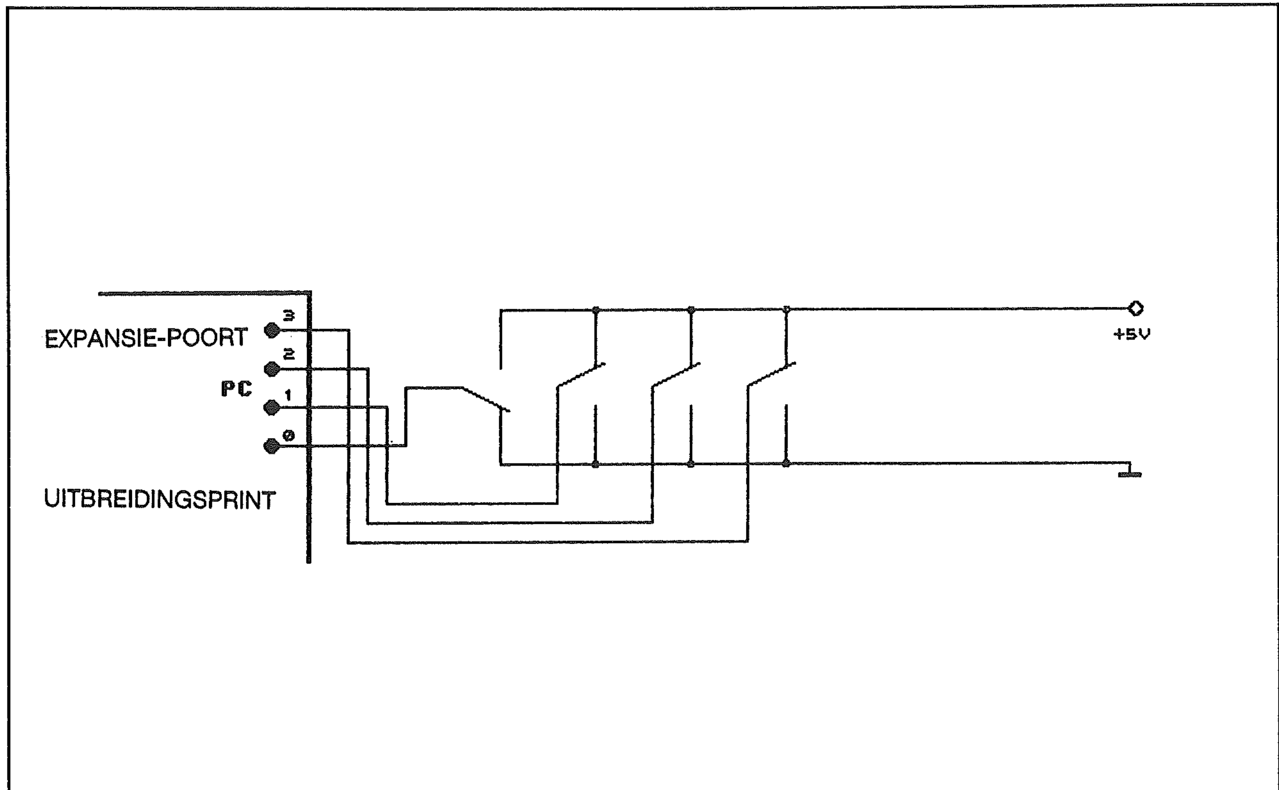
Figuur 4/6.4-12: Bit-voorbeeld en hexadecimale waarden.

Het bij deze schakeling horende programma is gegeven in listing 4/6.4-8.



Figuur 4/6.4-11: Testschakeling voor het expansiepoort-IC.

6.4 Expansiepoort voor de C 64 met 24 in- en uitgangen



Figuur 4/6.4-13: Testschakeling waarbij de lijnen PC0 tot en met PC3 als ingangen worden gebruikt.

Van regel C170 tot C178 worden alle poorten gewist. Na de sprong terug naar regel C155 worden de lijnen van poort C (PC0...PC3) opnieuw getest.

Als alle schakelaarcontacten aan 0 liggen, verschijnt er niets op het display. Omdat aan geen van de voorwaarden is voldaan loopt het programma door alle CMP en BEQ-regels van het programma.

Overzicht van de belangrijkste instructies en adresserings-methodes

Tot slot van dit hoofdstuk worden in de tabellen van de figuren 4/6.4-14 tot en met 4/6.4-21 een overzicht gegeven van de belangrijkste instructies.

6.4 Expansiepoort voor de C 64 met 24 in- en uitgangen

```
C100   A9 80     LDA  Laadt accu met stuurwoord 80(=24 uitgangen)
C102   8D 03 DE  STA  Bewaar accu inhoud in controlregister DE03
C105   A9 0F     LDA  Laadt accu met bit voorbeeld 0F.
C107   20 10 C0  JSR  Roep de poortsturings routine aan.
C10A   A9 F0     LDA  Laadt accu met bit voorbeeld F0.
C10C   20 10 C0  JSR  Roep de poortsturings routine aan.
C10F   A9 FF     LDA  bit voorbeeld FF
C111   20 10 C0  JSR
C114   A9 55     LDA  bit voorbeeld 55
C116   20 10 C0  JSR
C119   A9 AA     LDA  bit voorbeeld AA
C11B   20 10 C0  JSR
C11E   A9 00     LDA  bit voorbeeld 00
C120   20 10 C0  JSR
C123   4C 05 C1  JMP  Terug naar C105 om te herhalen.

Subroutine voor tijdsvertraging

C000   A2 FF     LDX  Laadt het X-register met FF
C002   A0 FF     LDY  Laadt het Y-register met FF
C004   88        DEY  Verminder Y-registerinhoud met 1.
C005   D0 FD     BNE  Als het Y-register nog niet nul is
                      spring dan naar C004.
C007   CA        DEX  Verminder X-registerinhoud met 1.
C008   D0 F8     BNE  Als het X-register nog niet nul is
                      spring dan naar C002.
C00A   60        RTS  Keer terug naar het aanroepende programma
Met dit programma wordt een tijdsvertraging bereikt van ongeveer
0,3 sec. Kortere tijdsvertragingen kunnen worden gerealsiseerd
door de waarden in C001 en C003 te verkleinen.


Subroutine voor poort-sturing

C010   8D 00 DE  STA  Sla accu inhoud op in poort A.
C013   8D 01 DE  STA  Sla accu inhoud op in poort B.
C016   8D 02 DE  STA  Sla accu inhoud op in poort C.
C019   20 00 C0  JSR  Roep de tijdsvertragingsroutine aan.
C01C   A9 00     LDA  Laadt accu met 00 (om poorten te resetten)
C01E   8D 00 DE  STA  poort A.
C021   8D 01 DE  STA  poort B.
C024   8D 02 DE  STA  poort C.
C027   20 00 C0  JSR  Roep de tijdsvertragingsroutine aan.
C02A   60        RTS  Keer terug naar het aanroepende programma.
```

Listing 4/6.4-7: Testprogramma 1.

6.4 Expansiepoort voor de C 64 met 24 in- en uitgangen

```
C150   A9 81     LDA  Laadt accu met stuurwoord 81
                      (4 ingangen / 20 uitgangen)
C152   8D 03 DE  STA  Bewaar accu inhoud in controlregister DE03
C155   AD 02 DE  LDA  Laad accu met bit-patroon van poort C
C158   C9 01     CMP  Compare accu met 01. Hiermee testen we of
                      schakelcontact PC0 hoog is, bitpatroon 0001
C15A   F0 22     BEQ  Is de accu inhoud gelijk aan 1? Ja: spring
                      naar C17E anders verder gaan met C15C.
                      CMP 01 trekt 01 van accu af zonder accu
                      inhoud te veranderen. Resultaat in vlag.
C15C   C9 02     CMP  Test of PC1 hoog is, bitpatroon 0010
C15E   F0 26     BEQ  Zo ja spring naar C186
C160   C9 04     CMP  Test of PC2 hoog is, bitpatroon 0100
C162   F0 2A     BEQ  Zo ja spring naar C18E
C164   C9 08     CMP  Test of PC3 hoog is, bitpatroon 1000
C166   F0 2E     BEQ  Zo ja spring naar C196
C168   C9 03     CMP  Combinatie test, bitpatroon 0011
C16A   F0 32     BEQ  Zo ja spring naar C19E
C16C   C9 0F     CMP  Bitpatroon 1111
C16E   F0 36     BEQ  Zo ja spring naar C1A6
C170   A9 00     LDA  Laadt accu met 00 voor wissen
C172   8D 00 DE  STA  Reset poort A
C175   8D 01 DE  STA  Reset poort B
C178   8D 02 DE  STA  Reset poort C
C17B   4C 55 C1  JMP  Spring terug naar C155
C17E   A9 0F     LDA  Bitpatroon 00001111
C180   20 30 C0  JSR  op de display zetten
C183   4C 55 C1  JMP  Spring terug naar C155
C186   A9 F0     LDA  Bitpatroon 11110000
C188   20 30 C0  JSR  op de display zetten
C18B   4C 55 C1  JMP  Spring terug naar C155
C18E   A9 81     LDA  Bitpatroon 10000001
C190   20 30 C0  JSR  op de display zetten
C193   4C 55 C1  JMP  Spring terug naar C155
C196   A9 CF     LDA  Bitpatroon 11001111
C198   20 30 C0  JSR  op de display zetten
C19B   4C 55 C1  JMP  Spring terug naar C155
C19E   A9 FF     LDA  Bitpatroon 11111111
C1A0   20 30 C0  JSR  op de display zetten
C1A3   4C 55 C1  JMP  Spring terug naar C155
C1A6   A9 55     LDA  Bitpatroon 01010101
C1A8   20 30 C0  JSR  op de display zetten
C1AB   4C 55 C1  JMP  Spring terug naar C155

Subroutine C030

C030   8D 00 DE  STA  Bewaar accu inhoud in poort A
C033   8D 01 DE  STA  Bewaar accu inhoud in poort B
C036   8D 02 DE  STA  Bewaar accu inhoud in poort C
C039   60        RTS  Keer terug naar het aanroepende programma
```

Listing 4/6.4-8: Testprogramma 2.

6.4 Expansiepoort voor de C 64 met 24 in- en uitgangen

Adresseringsmethode	B	LDA	LDX	LDY	STA	STX	STY
direkt	2	A9	A2	A0			
absoluut	3	AD	AE	AC	8D	8E	8C
zero-page	2	A5	A6	A4	85	86	84
absoluut x-ind.	3	BD		BC	9D		
absoluut y-ind.	3	B9	BE		99		
zero-page x-ind.	2	B5		B4	95		94
zero-page y-ind.	2		B6			96	
indirekt indexed	2	B1			91		
indexed indirekt	2	A1			81		

De kolom B geeft het aantal bytes per instruktie.
ind. = indexed

Figuur 4/6.4-14: De laad- en bewaar-instructies.

Adresseringsmethode	B	TAX	TAY	TXA	TYA	TXS	TSX
impliciet	1	AA	A8	8A	98	9A	BA

Figuur 4/6.4-15: De trasfer-instructies.

Adresseringsmethode	B	ADC	SBC	INC	DEC	INX	DEX	INY	DEY
direkt	2	69	E9						
absoluut	3	6D	ED	EE	CE				
zero-page	2	65	E5	E6	C6				
absoluut x-ind.	3	7D	FD	FE	DE				
absoluut y-ind.	3	79	F9						
zero-page x-ind.	2	75	F5	F6	D6				
indirekt indexed	2	71	F1						
indexed indirekt	2	61	E1						
impliciet	1					E8	CA	C8	88

Figuur 4/6.4-16: De rekenkundige instructies.

6.4 Expansiepoort voor de C 64 met 24 in- en uitgangen

Adresseringsmethode	B	AND	ORA	EOR	CMP	CPX	CPY	BIT
direkt	2	29	09	49	C9	E0	C0	
absoluut	3	2D	0D	4D	CD	EC	CC	2C
zero-page	2	25	05	45	C5	E4	C4	24
absoluut x-ind.	3	3D	1D	5D	DD			
absoluut y-ind.	3	39	19	59	D9			
zero-page x-ind.	2	35	15	55	D5			
indirekt indexed	2	31	11	51	D1			
indexed indirekt	2	21	01	41	C1			

Figuur 4/6.4-17: De logische en vergelijkings-instructies.

Adresseringsmethode	B	BCC	BCS	BEQ	BNE	BMI	BPL	BVC	BVS
relatief	2	90	B0	F0	D0	30	10	50	70

Figuur 4/6.4-18: De vertakkings-instructies.

Adresseringsmethode	B	JSR	JSR
absoluut	3	4C	20
indirekt	3	6C	

Figuur 4/6.4-19: De sprong-instructies.

6.4 Expansiepoort voor de C 64 met 24 in- en uitgangen

Adresseringsmethode	B	ASL	LSR	ROL	ROR
accu	1	0A	4A	2A	6A
absoluut	3	0E	4E	2E	6E
zero-page	2	06	46	26	66
absoluut x-ind.	3	1E	5E	3E	7E
zero-page x-ind.	2	16	56	36	76

Figuur 4/6.4-20: De schuif-instructies.

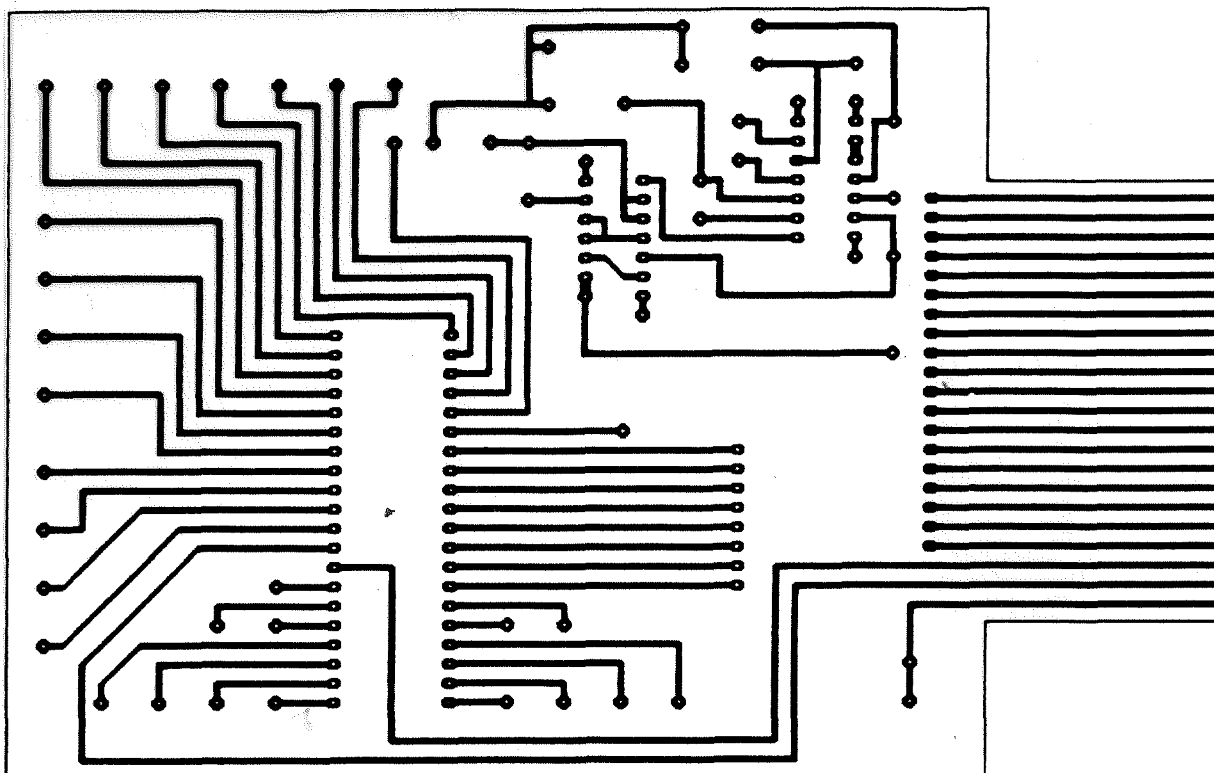
Statusregister	B	CLC	CLD	CLI	CLV	SEC	SED	SEI
impliciet	1	18	D8	58	B8	38	F8	78

Stack-instrukties	B	PLA	PHA	PLP	PHP
impliciet	1	68	48	28	08

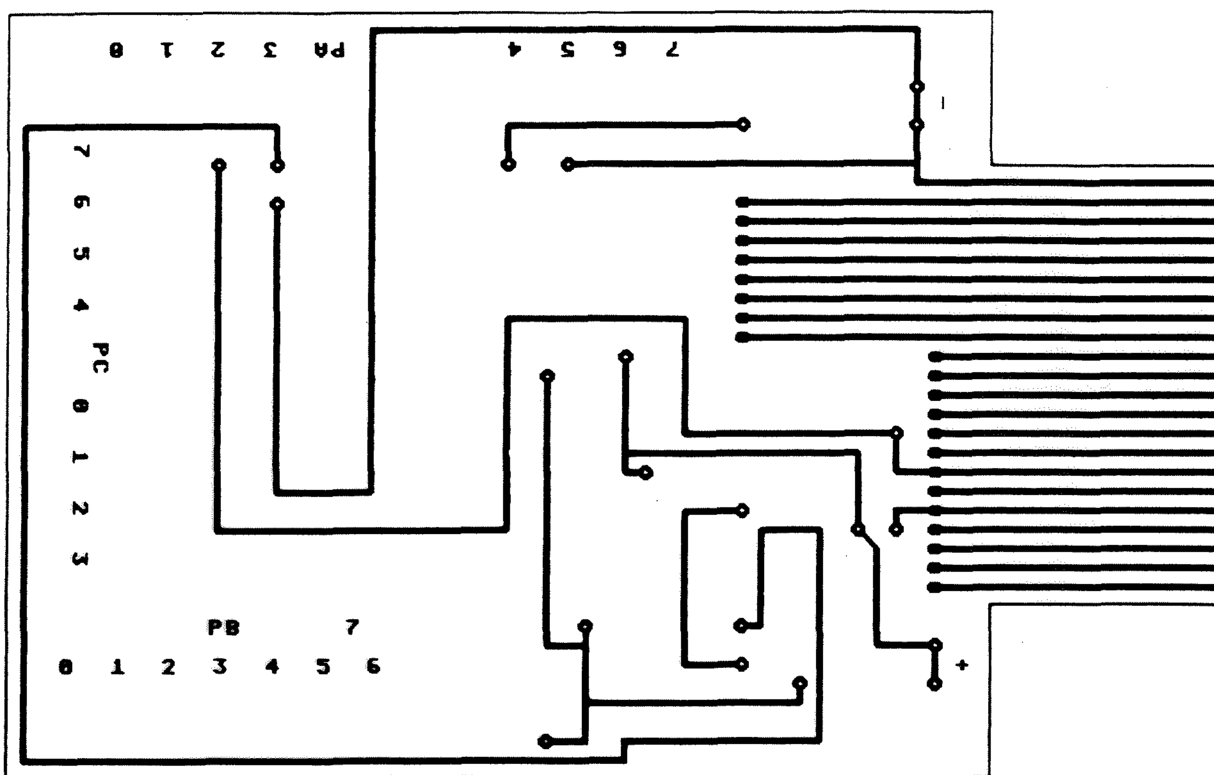
Diversen	B	NOP	RTS	RTI	BRK
impliciet	1	EA	60	40	00

Figuur 4/6.4-21: Het status-register, de stack en de overige instructies.

6.4 Expansiepoort voor de C 64 met 24 in- en uitgangen



Figuur 4/6.4-8 en figuur 4/6.4-9: De twee zijden van de dubbelzijdige print



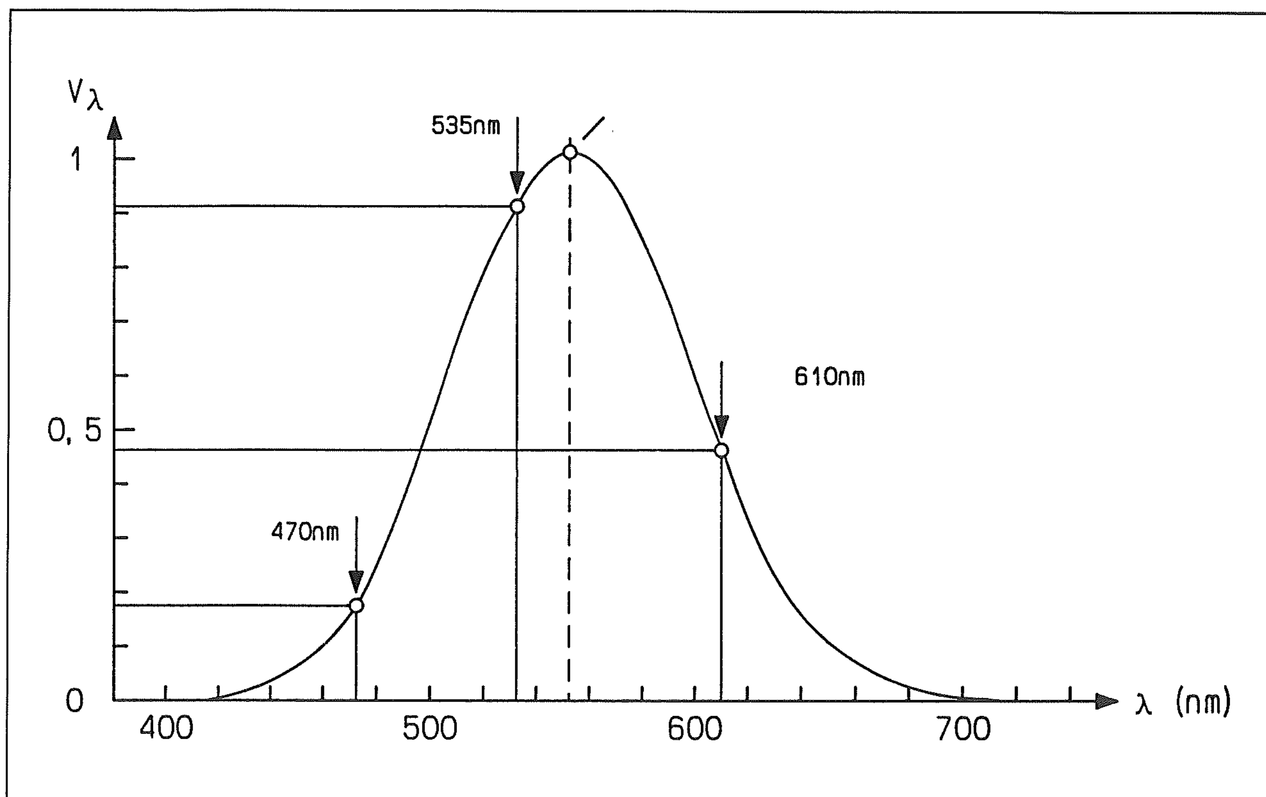
4/6.5

RGB naar composite video omzetter

Inleiding
De meeste computers zijn tegenwoordig voorzien van een kleurenkaart, waarbij de drie kleursignalen R, G en B afzonderlijk worden uitgevoerd. Sommige kaarten leveren digitale RGB-signalen, andere analoge. De digitale kaarten hebben bovendien vaak een afzonderlijke I-uitgang, waarmee men de intensiteit van een digitale monitor kan halveren. Daarnaast staan ook nog eens een of twee sync-signalen ter beschikking.
Met deze ingewikkelde combinatie van signalen is het niet zonder meer mogelijk een oude analoge monitor met composite video-ingang aan te sturen. Toch zou het interessant zijn als dat wél kon, want vele computeraars hebben nog wel ergens zo'n oude zwart/wit monitor, vaak met een groot scherm, ongebruikt staan. Voor bepaalde toepassingen, bijvoorbeeld klassikaal onderwijs of instructie, is het handig het beeld van de 14 inch standaard kleuren-monitor parallel op zo'n veel grotere beeldbuis te vertonen, al is het dan in zwart/wit.
Met de in dit hoofdstuk voorgestelde schakeling kan dat.
De schakeling zet de afzonderlijke signalen van een CGA-, EGA-, VGA-, PGA- en MCGA-kaart om in een composite videosignaal. Bovendien levert de schakeling nog eens een aantal extra sync-signalen. Daarnaast kan men de schakeling ook toepassen bij ST-computers van Atari en bij de Amiga's van Commodore.

Beperkingen
Er zijn natuurlijk beperkingen aan de omzetting van de genoemde systemen. Moderne monitoren werken met multi-sync. Dat wil zeggen dat de monitor zowel voor de horizontale als voor de vertikale afbuigfrequenties vrij grote marges heeft. Welke frequenties worden gebruikt hangt af van de grafische modus die men op de grafische kaart instelt.
Zo kan men vele SVGA-kaarten instellen op een vertikale sync van 80 Hz. Het zal duidelijk zijn dat de meeste composite video-monitoren, met hun eigen sync van 47 tot 63 Hz, die snelheid niet kunnen bijbenen!
Hetzelfde geldt uiteraard voor de horizontale sync. Een standaard monitor werkt met een sync-frequentie van 15,625 tot 18,432 kHz. Het zal alweer duidelijk zijn dat de lijngenerator in een dergelijke apparaat moderne sync's van meer dan 70 kHz niet kan verwerken.
Tot slot is er nog het probleem van de bandbreedte. Standaard video-monitoren hebben een bandbreedte van 20 MHz. Moderne VGA-systemen werken echter met bandbreedtes tot 50 MHz! Composite video-monitoren kunnen deze bandbreedtes niet goed weergeven en er zullen beeldvervormingen ontstaan.

6.5 RGB naar composite video omzetter



Figuur 4/6.5-1: De gevoeligheidscurve van het menselijke oog.

Men zal dus, na het bouwen van de schakeling, moeten experimenteren welke modus van de grafische kaart geschikt is voor het aansturen van een composite video-monitor. Men moet bovendien niet verwachten dat op het grote beeldscherm hetzelfde scherpe beeld ontstaat dat men gewend is van moderne VGA-monitoren! Ondanks deze beperkingen kan het bouwen van de beschreven schakeling, zeker omdat zij zo goedkoop is, voor sommige toepassingen toch zinvol zijn.

Kleurentheorie

Wie nu denkt dat het volstaat de afzonderlijke uitgangen voor rood, groen en blauw met elkaar te mengen heeft het mis! Er zou een zeer vreemdsoortig grijsbeeld ontstaan, waarin bepaalde grijstinten zouden overheersen.

Dit is een gevolg van de nogal beperkte bandbreedte van het menselijke oog. De gevoeligheidscurve van het menselijke oog in functie van de kleur van het licht (de golflengte in nm) is voorgesteld in figuur 4/6.5-1.

Uit deze grafiek blijkt duidelijk dat het oog het gevoeligst is voor licht met een golflengte van 555 nm. Dit licht heeft een kleur die ergens tussen geel en groen ligt. Voor rood (610 nm) en blauw (470 nm) is de gevoeligheid van het oog erg laag. Het gevolg van een en ander is dat de drie R-, G- en B-signalen niet zonder meer met elkaar gemengd mogen worden, maar via specifieke verzwakkers.
Als men de maximale gevoeligheid van het oog door 1 voorstelt, kan men uit de grafiek berekenen dat de gevoeligheden

6.5 RGB naar composite video omzetter

voor de drie basiskleuren rood, groen en blauw gelijk zijn aan:
- 0,92 voor groen;
- 0,47 voor rood;
- 0,17 voor blauw.

Uit deze drie gegevens kan men de formule afleiden, waaraan de menger moet voldoen. De totale versterking wordt gelijkgesteld aan 1. Om nu de drie deelgevoeligheden naar deze eenheid om te rekenen worden de drie gevoeligheidswaarden opgeteld.
Dus:
- basisgevoeligheid:
  0,92 + 0,47 + 0,17 = 1,56
- gevoeligheid voor rood:
  0,47/1,56 = 0,3
- gevoeligheid voor blauw:
  0,17/1,56 = 0,11
- gevoeligheid voor groen:
  0,92/1,56 = 0,59

Aan de hand van deze drie factoren kan men de formule samenstellen, die het verband geeft tussen de amplitude van het Y-signaal (het monochroom videosignaal) en de drie ter beschikking staande signalen R, G en B:
Y = 0,3R + 0,59G + 0,11B
Deze formule komt, elektronisch bekeken, eigenlijk overeen met een passieve menger, die samengesteld kan worden uit een weerstandsmatrix.

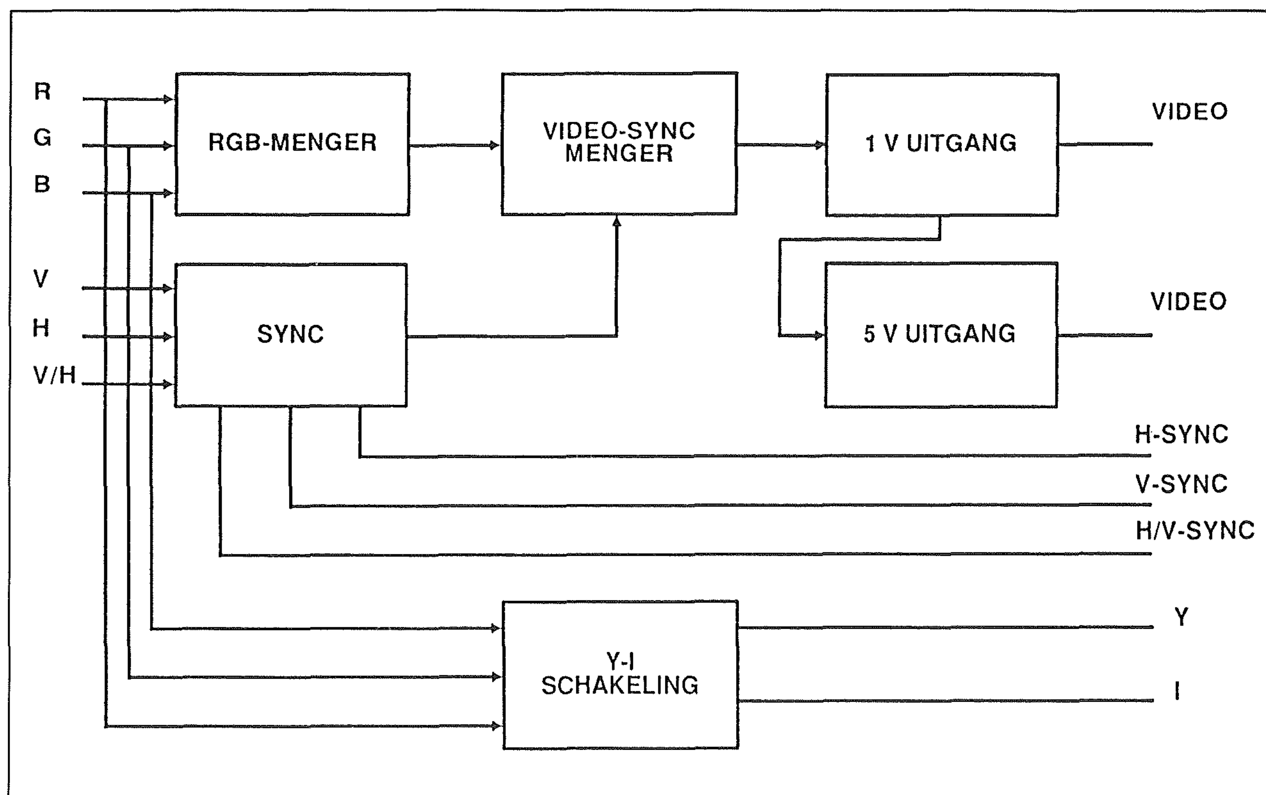
Blokschema van de omzetter
Het blokschema van de omzetter is getekend in figuur 4/6.5-2.
De drie kleursignalen R, G en B worden aan de in principe reeds beschreven RGB-menger aangeboden. Deze genereert een video-signaal, waar rekening wordt gehouden met de gevoeligheid van het oog. Nadien volgt de video-sync menger. In deze trap wordt het horizontale en vertikale sync-signaal met het video-signaal gemengd, zodat een normaal composite video-signaal ontstaat. Dit is grafisch toegelicht in figuur 4/6.5-3.

Dat sync-signaal is afkomstig uit het sync-blok. Aan dit blok kan men de horizontale, vertikale of gemengde sync-signalen van de grafische kaart aanbieden. Het blok mengt deze signalen en heeft bovendien de mogelijkheid te kiezen tussen positieve of negatieve sync-pulsen. Op deze manier zal er wel geen grafisch systeem bestaan, dat niet door deze schakeling verwerkt kan worden! Het sync-blok levert bovendien nog eens drie afzonderlijke sync-uitgangen op TTL-niveau, zodat ook monitoren met afzonderlijke sync-aansluitingen te sturen zijn.

Het composite video-signaal gaat naar twee uitgangsversterkers. De ene is afgeregeld op het standaard 1 V niveau van de meeste monitoren. De tweede versterker pept het composite video-signaal op tot 5 V, want er schijnen ook monitoren te bestaan die met dit niveau werken.
Het onderste blok, de Y-I schakeling, maakt de drie kleursignalen geschikt voor het aansturen van de zogenoemde Y-I monitoren. Deze monitoren werken digitaal en hebben afzonderlijke ingangen voor video (Y) en intensiteit (I). Met deze laatste ingang kan de intensiteit tot de helft worden terug gebracht. De beschreven schakeling heeft draadbruggen, waarmee men kan instellen welke basiskleuren op de volle intensiteit en welke basiskleuren op de halve intensiteit worden weergegeven. Met deze monitoren zijn dus geen grijstinten weer te geven!

Het volledige schema
Het volledige schema van de schakeling is getekend in figuur 4/6.5-4.

6.5 RGB naar composite video omzetter

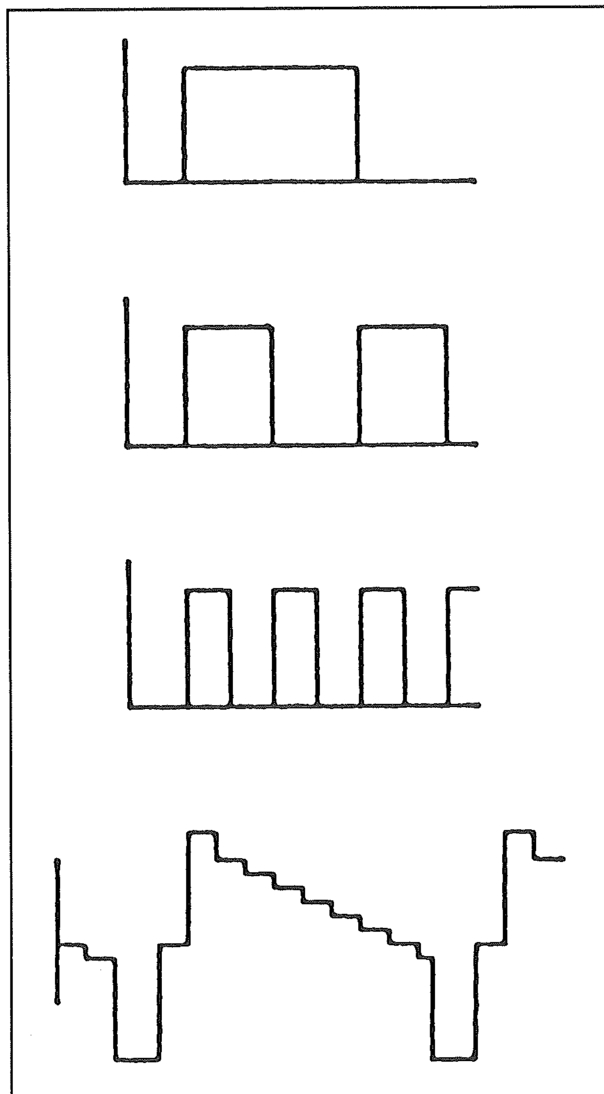


Figuur 4/6.5-2: Het blokschema van de RGB naar composite video omzetter.

De drie kleursignalen worden aangeboden aan de klemmen ST1, ST2 en ST3. De weerstanden R1 tot en met R6 vormen de matrix, die de vergelijking van het Y-signaal uitvoert. De drie draadbruggen BR14, BR16 en BR17 moeten aangebracht worden als de drie ingangssignalen een topwaarde van 1 V hebben. Zonder deze draadbruggen verwacht de schakeling TTL-compatible ingangssignalen. Het laatste is het geval als de kleurenkaart digitaal werkt, het eerste als men een analoge kaart heeft. Het aldus gegenereerde video-signaal wordt via weerstand R21 gemengd met het composite sync-signaal.

Het volledige composite video-signaal wordt via de scheidingscondensator C1 aan de eerste eindversterker aangeboden. Dit is een direct gekoppelde tweetraps versterker, opgebouwd rond de transistoren T1 en T2. De versterking van de schakeling wordt bepaald door de verhouding tussen de weerstanden R10 en R11. De condensator C2 zorgt voor het recht trekken van de bandbreedte aan de hoge kant. De gelijkspanningsinstelling van de tweetraps versterker wordt verzorgd door de weerstanden R8 en R9. De versterking van de schakeling is zo berekend dat aan de uitgang ST10 een composite video-signaal met een top-tot-top waarde van 1 V ter beschikking staat. De meeste monitoren kunnen een dergelijk signaal zonder problemen verwerken.
De uitgangsimpedantie van de versterker wordt bepaald door de weerstand R12 en bedraagt dus 75 Ω.
Het signaal op de collector van T2 wordt via de scheidingscondensator C8 aangeboden aan de ingang van de tweede uitgangsversterker.

6.5 RGB naar composite video omzetter



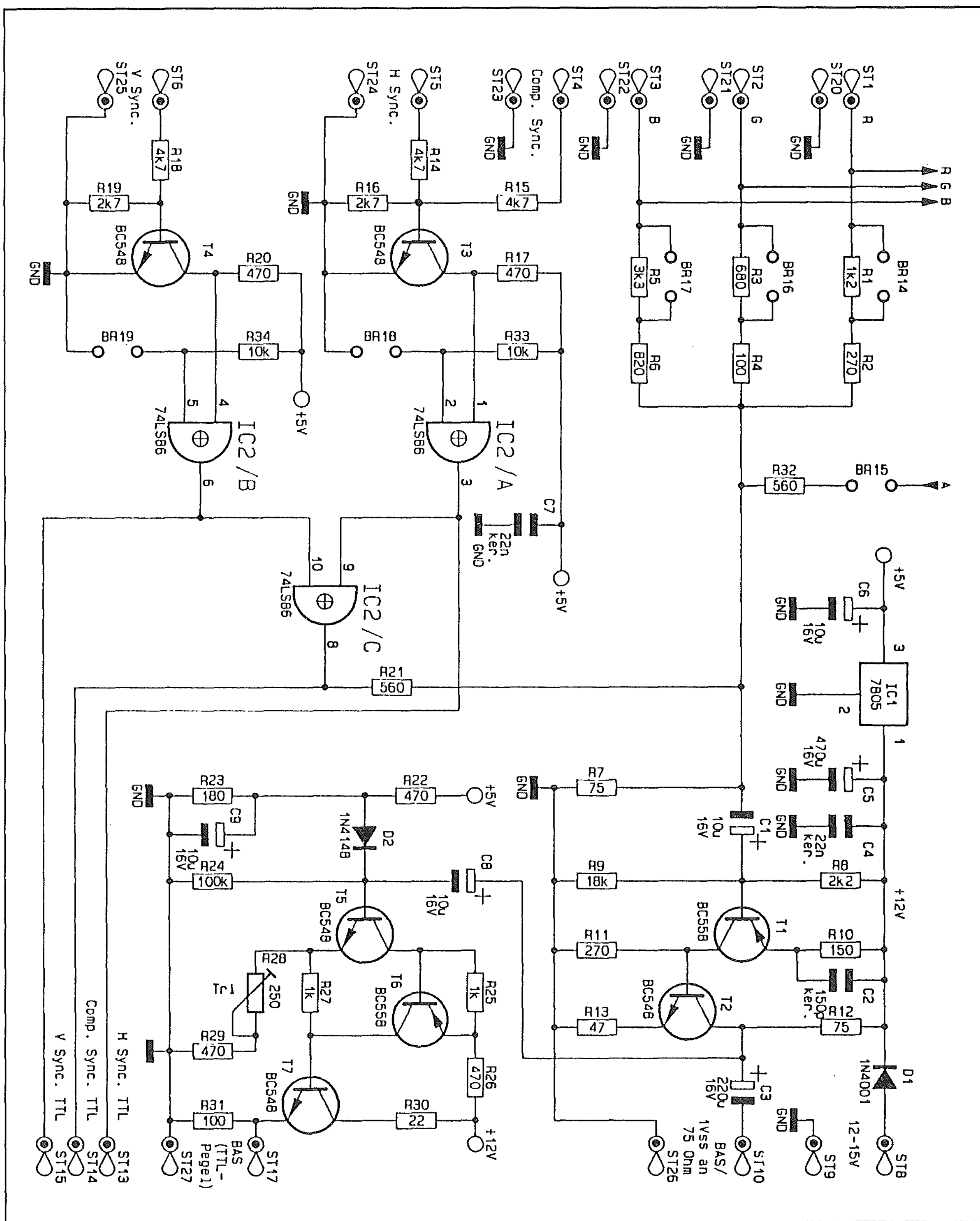
Figuur 4/6.5-3: De omzetting van de vier signalen R, G, B en SYNC (boven) in een composite video-signaal (onder).

Deze levert een uitgangssignaal met een top-tot-top waarde van 5 V. Deze versterker wordt ingesteld door de weerstandsdeler R22/R23. Het video-signaal wordt via de diode D2 op deze instelspanning geclampt. Op deze manier is men er zeker van dat geen gelijkspanningsverschuivingen optreden en dat de zwart- en witdrempels gehandhaafd blijven

De tweede versterker bestaan uit de transistoren T5, T6 en T7. Hierbij werken T5 en T6 als versterker en T7 als emittervolger. De versterking van de trap kan ingesteld worden met de instelpotentiometer R28. De top-tot-top waarde van het uitgangssignaal kan met dit onderdeel ingesteld worden tussen 4 en 6 V. De collectorweerstand R30 zorgt voor een stabiele, oscillatievrije werking van de uitgangstrap. Het 5 V signaal wordt rechtstreeks van de emitter van T7 afgenomen via ST17.
Denk er aan dat deze uitgang zeer laagimpedant is en niet kortsluitvast!

Als de grafische kaart een composite syncsignaal levert moet dit worden aangesloten op ST4. Werkt de kaart met gescheiden sync-pulsen, dan moet de V-sync op ST6 en de H-sync op ST5 worden aangesloten. De trappen rond de transistoren T3 en T4 zorgen voor een impedantie- en niveau-aanpassing naar TTL. De trappen zijn in staat sync-pulsen te verwerken met een amplitude tussen 1 V en 5 V.
Met behulp van de draadbruggen BR18 en BR19 kan men de schakeling instellen op het verwerken van positieve of negatieve sync-pulsen. Als de draadbruggen niet aanwezig zijn leveren de uitgangen van de poorten IC2/A en IC2/B sync-pulsen met dezelfde polariteit als deze van de ingangspulsen. Met aangebrachte draadbruggen leveren de poorten geïnverteerde pulsen af.
De twee poortuitgangen gaan rechtstreeks naar de uitgangen ST13 en ST15, waar dus sync-signalen op TTL-niveau ter beschikking staan.
De poort IC2/C zorgt voor het genereren van het composite sync-signaal. In deze poort worden de horizontale en vertikale sync-pulsen gecombineerd.

6.5 RGB naar composite video omzetter



Figuur 4/6.5-4: Het volledige schema van de RGB-naar-video omzetter.

6.5 RGB naar composite video omzetter

Het uitgangssignaal wordt op de reeds beschreven manier via weerstand R21 gemengd met het video-signaal. Ook deze uitgang wordt aan een aansluitpen aangeboden en wel aan ST14.

De voeding
De schakeling kan gevoed worden met een positieve gelijkspanning tussen 12 V en 15 V. De diode D1 zorgt ervoor dat de schakeling niet defect gaat als men per ongeluk de voeding verkeerd om aansluit. De twee eindversterkers worden rechtstreeks gevoed uit deze spanning. Voor de digitale schakelingen staat de stabilisator IC1 ter beschikking, die uit de 12 V tot 15 V een 5 V gelijkspanning genereert.

De Y-I schakeling
Deze schakeling is getekend in figuur 4/6.5-5. De drie primaire kleuren R, G en B moeten nu wel als digitaal signaal ter beschikking staan. Met de draadbruggen BR1 tot en met BR6 kan men het rode, groene en/of blauwe signaal selecteren voor de aansturing op volle intensiteit van de Y-I monitor. Met de draadbruggen BR7 tot en met BR13 kan men twee kleuren kiezen, die de monitor met halve helderheid zullen aansturen. De I-ingang van de monitor wordt aangesloten op ST11, de Y-ingang op ST12. Sommige grafische kaarten, zoals de oude CGA, hebben een rechtstreekse I-uitgang. In dat geval moet dit signaal op ST7 aangesloten worden en wordt de I-ingang van de monitor verbonden met ST16.
Een paar voorbeelden.
Stel dat men wil dat de kleuren rood en groen met volle intensiteit worden weergegeven en de kleur blauw met de halve intensiteit. Men moet dan de draadbruggen BR1, BR3, BR6, BR9 en BR12 op de print aanbrengen. Wil men dat alleen groen met volle intensiteit wordt weergegeven en rood en blauw met halve intensiteit, dan moeten de draadbruggen BR2, BR3, BR6, BR7 en BR12 gesoldeerd worden.

Van digitaal RGB naar analoog video
Sommige kleurenkaarten leveren digitale RGB-signalen. De schakeling mengt deze signalen tot een analoog composite videosignaal, waarmee analoge monitoren aangestuurd kunnen worden. Op deze manier ontstaan acht grijstinten. Heeft de grafische kaart bovendien nog eens een I-uitgang, dan moet men deze aansluiten op ST7 en de draadbrug BR15 aanbrengen. Het I-signaal wordt dan via de weerstand R32 gemengd met de drie digitale RGB-signalen, zodat in totaal 16 grijstinten gegenereerd worden.

Onderdelenlijst

Weerstanden, 1/4 W:

R30	=	22	Ω
R13	=	47	Ω
R7,R12	=	75	Ω
R4,R31	=	100	Ω
R10	=	150	Ω
R23	=	180	Ω
R2,R11	=	270	Ω
R17,R20,R22, R26,R29	=	470	Ω
R21,R32	=	560	Ω
R3	=	680	Ω
R6	=	820	Ω
R25,R27	=	1	kΩ
R1	=	1,2	kΩ
R8	=	2,2	kΩ
R16,R19	=	2,7	kΩ
R5	=	3,3	kΩ
R14,R15,R18	=	4,7	kΩ
R33,R34	=	10	kΩ
R9	=	18	kΩ

6.5 RGB naar composite video omzetter

R24	=	100 kΩ	

Instelpotentiometer, staand:

R28	=	250 Ω	

Condensatoren:

C2	=	150 pF	ceramisch
C4,C7	=	22 nF	ceramisch
C1,C6,C8,C9	=	10 µF	16 V elco
C3	=	220 µF	16 V elco
C5	=	470 µF	16 V elco

Halfgeleiders:

D1	=	1N4001
D2	=	1N4148
T1,T6	=	BC558
T2,T3,T4,T5,T7	=	BC548
IC1	=	7805
IC2	=	74LS86
IC3	=	74HC32

De bouw van de schakeling

De volledige schakeling kan ondergebracht worden op een klein printje met als afmetingen 64 x 86 mm^2. Het ontwerp is getekend als figuur 4/6.5-6 op de transparante printpagina. De componentenopstelling is getekend in figuur 4/6.5-7.
In tegenstelling tot wat de normale volgorde is, worden bij deze schakeling de talloze draadbruggen tot het einde bewaard. Nadat alle onderdelen zijn aangebracht bepaalt men aan de hand van de grafische kaart waarover men beschikt welke draadbruggen aangebracht moeten worden. Nadien kan de externe bedrading naar de uitgangsconnector van de grafische kaart en de ingangsconnector van de monitor worden uitgevoerd.
In principe is het mogelijk alle draadbruggen te vervangen door mini-DIP schakelaartjes. Dat is natuurlijk alleen zinvol als men de schakeling op verschillende systemen wil beproeven.

De uitgangsconnectoren van video-systemen

Tot slot van deze bouwbeschrijving is het noodzakelijk de uitgangssignalen van de beschikbare video-kaarten in het kort te bespreken.

De CGA-standaard voor PC's

Deze maakt gebruik van een 9-polige Sub-D connector, geschetst in figuur 4/6.5-8.

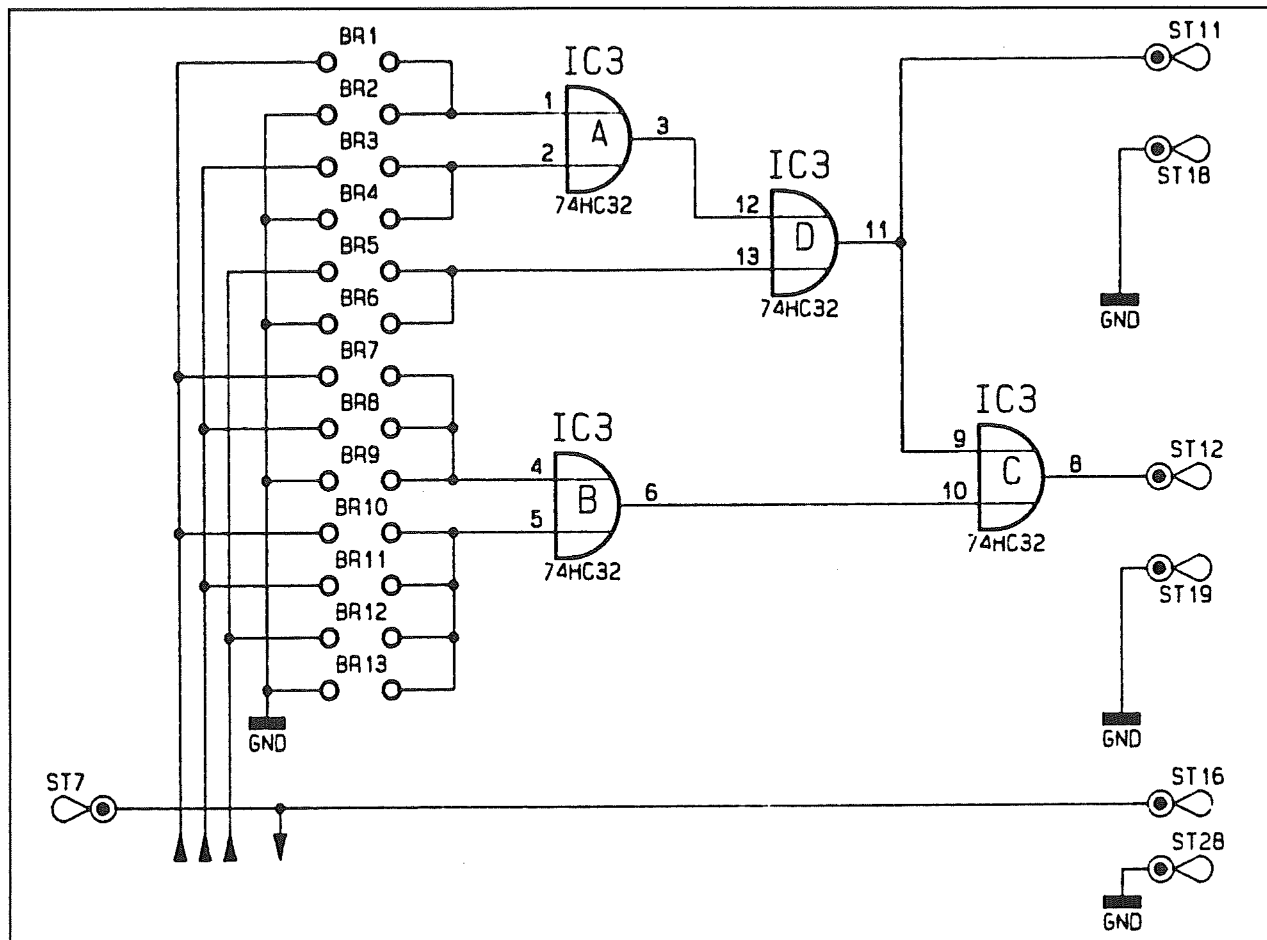
De aansluitingen van de uitgangssignalen op de connector zijn als volgt:
- pen 1: massa;
- pen 2: massa;
- pen 3: digitaal rood;
- pen 4: digitaal groen;
- pen 5: digitaal blauw;
- pen 6: digitaal intensiteit;
- pen 7: niet aangesloten;
- pen 8: positieve H-sync;
- pen 9: negatieve V-sync.

De EGA-standaard voor PC's

Ook deze kaarten maken gebruik van een 9-polige Sub-D connector. Voor iedere kleur staan nu echter twee digitale uitgangen ter beschikking, een die het minst belangrijke bit levert en een die het meest belangrijke bit levert. Men moet de MSB's op de ingangen van de omzetter aansluiten.

De aansluitingen van de uitgangssignalen op de connector zijn als volgt:
- pen 1: massa;
- pen 2: digitaal rood LSB;
- pen 3: digitaal rood MSB;
- pen 4: digitaal groen MSB;
- pen 5: digitaal blauw MSB;
- pen 6: digitaal groen LSB;
- pen 7: digitaal blauw LSB;
- pen 8: positieve H-sync;
- pen 9: negatieve V-sync.

6.5 RGB naar composite video omzetter



Figuur 4/6.5-5: De schakeling voor het instellen van de Y-I uitgangen.

De PGC-standaard voor PC's

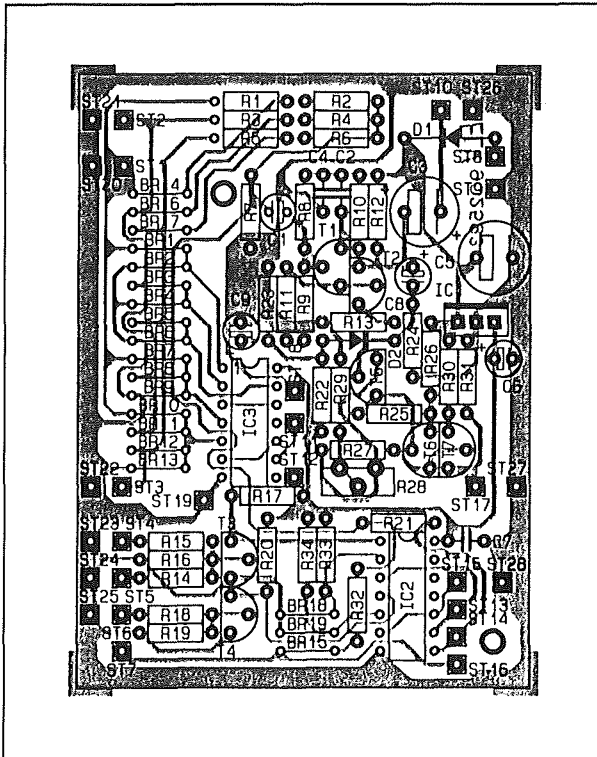
Ook deze kaarten hebben een 9-polige Sub-D connector met de volgende aansluitingen:
- pen 1: digitaal rood;
- pen 2: digitaal groen;
- pen 3: digitaal blauw;
- pen 4: composite sync;
- pen 5: mode;
- pen 6: rode massa;
- pen 7: groene massa;
- pen 8: blauwe massa;
- pen 9: algemene massa.

De MCGA-standaard voor PC's

Deze door IBM geïntroduceerde standaard werkt met een 15-polige Sub-D connector, getekend in figuur 4/6.5-9.

De MCGA-standaard levert analoge RGB-signalen af, die door de in dit hoofdstuk beschreven schakeling gemengd worden en goed zijn voor 256 grijstinten op het scherm van een analoge composite zwart/wit video-monitor.
De aansluitingen van de uitgangssignalen op de connector zijn als volgt:

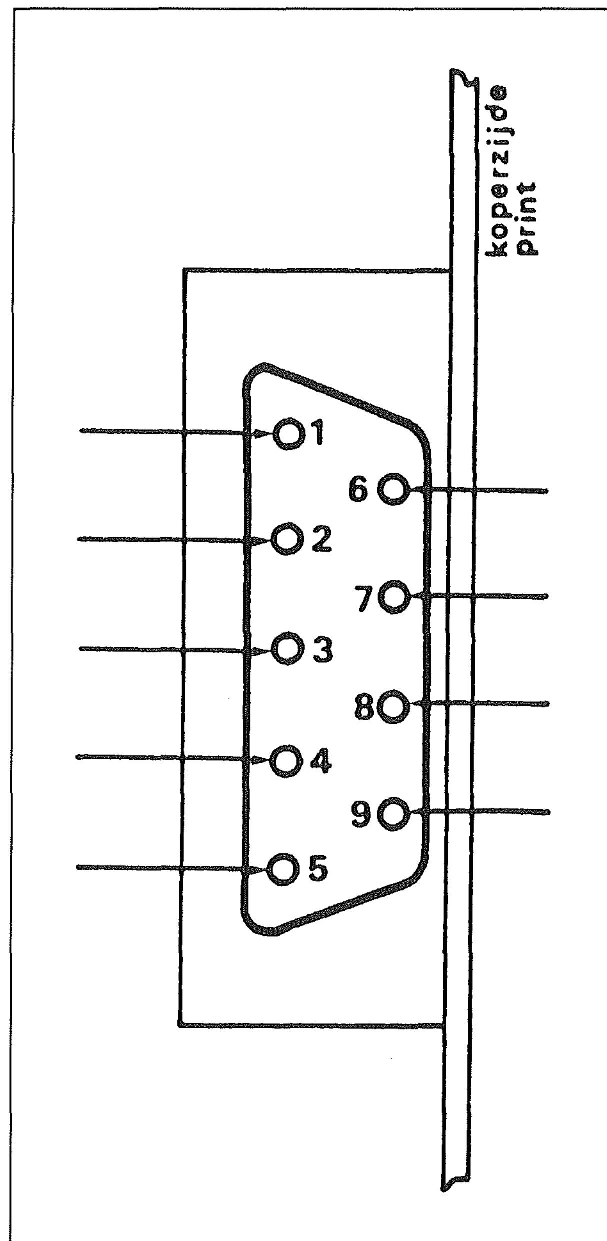
6.5 RGB naar composite video omzetter



Figuur 4/6.5-7: De componentenopstelling van de RGB-naar-video omzetter.

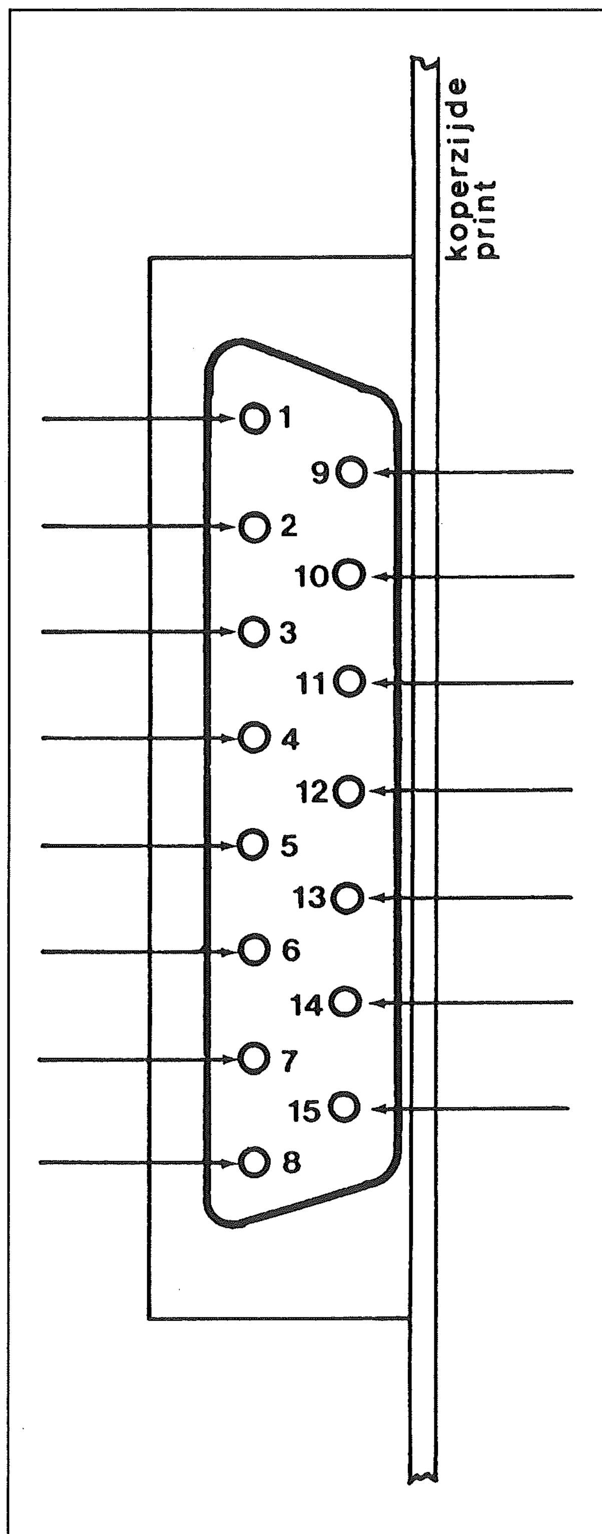
- pen 1: analoog rood;
- pen 2: analoog groen;
- pen 3: analoog blauw;
- pen 4: niet aangesloten;
- pen 5: test;
- pen 6: rode ingang;
- pen 7: groene ingang;
- pen 8: blauwe ingang;
- pen 9: key-ingang;
- pen 10: massa;
- pen 11: type 0;
- Pen 12: type 1;
- Pen 13: H-sync;
- pen 14: V-sync;
- pen 15: niet aangesloten.

De speciale pennen key, type en test worden gebruikt om de kaart te testen en het soort monitor dat op de kaart is aangesloten te identificeren. Omdat bij deze schakeling de zwart/wit composite videomonitor parallel aan de bestaande monitor wordt aangesloten, heeft men niets te maken met deze signalen.



Figuur 4/6.5-8: De 9-polige Sub-D connector van CGA, EGA en PGC.

6.5 RGB naar composite video omzetter



Figuur 4/6.5-9: De 15-polige Sub-D connector die wordt gebruikt bij MCGA en VGA.

De VGA-standaard voor PC's
Deze moderne standaard werkt eveneens met de 15-polige Sub-D connector, getekend in figuur 4/6.5-9.
Ook de VGA-standaard werkt analoog.
De aansluitingen van de uitgangssignalen op de connector zijn als volgt:
- pen 1: analoog rood;
- pen 2: analoog groen;
- pen 3: analoog blauw;
- pen 4: monitor identificatie;
- pen 5: algemene massa;
- pen 6: rode massa;
- pen 7: groene massa;
- pen 8: blauwe massa;
- pen 9: niet aangesloten;
- pen 10: sync massa;
- pen 11: monitor identificatie;
- Pen 12: monitor identificatie;
- Pen 13: H-sync;
- pen 14: V-sync;
- pen 15: niet aangesloten.

Opgemerkt moet worden dat standaard VGA-kaarten met een horizontale frequentie van 60 tot 70 kHz werken en dat het dus maar de vraag is of composite video-monitoren deze signalen kunnen verwerken!

De Atari ST-standaard
Deze computers hebben een ronde, 13-polige connector, waarvan de pencodering getekend is in figuur 4/6.5-10.

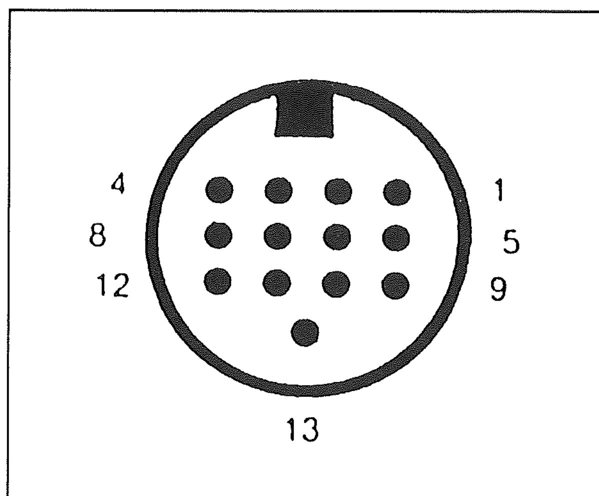
De aansluitingen van de uitgangssignalen op de connector zijn als volgt:
- pen 1: audio-uitgang;
- pen 2: composite sync;
- pen 3: geluidsgenerator;
- pen 4: monochroom sensor;
- pen 5: audio-ingang;
- pen 6: groene uitgang;
- pen 7: rode uitgang;
- pen 8: over 1,2 kΩ aan +12 V;

6.5 RGB naar composite video omzetter

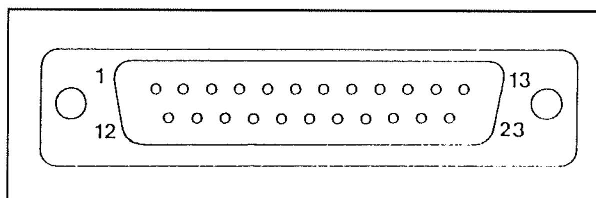
- pen 9: H-sync;
- pen 10: blauwe uitgang;
- pen 11: monochroom signaal;
- Pen 12: V-sync;
- Pen 13: massa.

De Amiga's van Commodore
Deze computers hebben een 23-polige video-connector, getekend in figuur 4/6.5-11.

De Amiga levert zowel analoge als digitale RGB-signalen. Bovendien staat op een pen van de connector de interne voedingsspanning van +12 V ter beschikking. Deze uitgang kan voldoende vermogen leveren voor het voeden van de beschreven schakeling.
De voor deze toepassing belangrijke pennen:
- pen 3: analoog rood;
- pen 4: analoog groen;
- pen 5: analoog blauw;
- pen 6: digitaal I-signaal;
- pen 7: digitaal blauw;
- pen 8: digitaal groen;
- pen 9: digitaal rood;
- pen 10: composite sync;
- pen 11: H-sync;
- pen 12: V-sync;
- pennen 16 - 20: massa;
- pen 22: +12 V voedingsspanning.

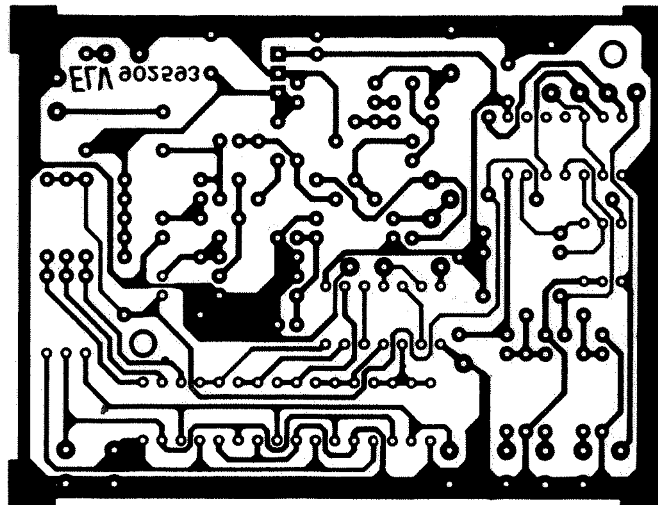


Figuur 4/6.5-10: De aansluitcodering van de video-connector van ST-computers.



Figuur 4/6.5-11: De aansluitcodering van de 23-polige connector bij Amiga computers.

6.5 RGB naar composite video omzetter



Figuur 4/6.5-6: De print van de schakeling.

4/6.6

Optisch geïsoleerde RS232 interface

Inleiding

Alle computers die serieel met de buitenwereld willen communiceren bedienen zich van het internationaal gestandaardiseerde RS232C protocol, ook wel V24 genoemd. Deze interface dient al lang niet meer alleen voor het besturen van een muis en een modem. Allerlei besturingssystemen kunnen via een V24 interface met de computer verbonden worden. Zo zijn er systemen in de handel waarmee men elektronische meetwaarden als temperatuur, druk, etc. via een RS232C kabel met de seriële poort van een computer kan verbinden. De seriële interface heeft een groot aantal pluspunten. Zo is bidirectionele data-transfer mogelijk, waarbij de gegevens over slechts twee aders, TXD en RXS genoemd, heen en weer worden gezonden. Er bestaan eenvoudige, volledig softwaregestuurde protocollen, die het data-transport regelen. De "L"- en "H"-niveaus zijn internationaal gestandaardiseerd en de storingsongevoeligheid is groot.

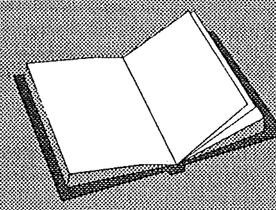
Helaas heeft de seriële interface ook een aantal nadelen. De voornaamste zijn:

- massa-lussen;
- paracitaire capaciteiten;
- elektromagnetische straling.

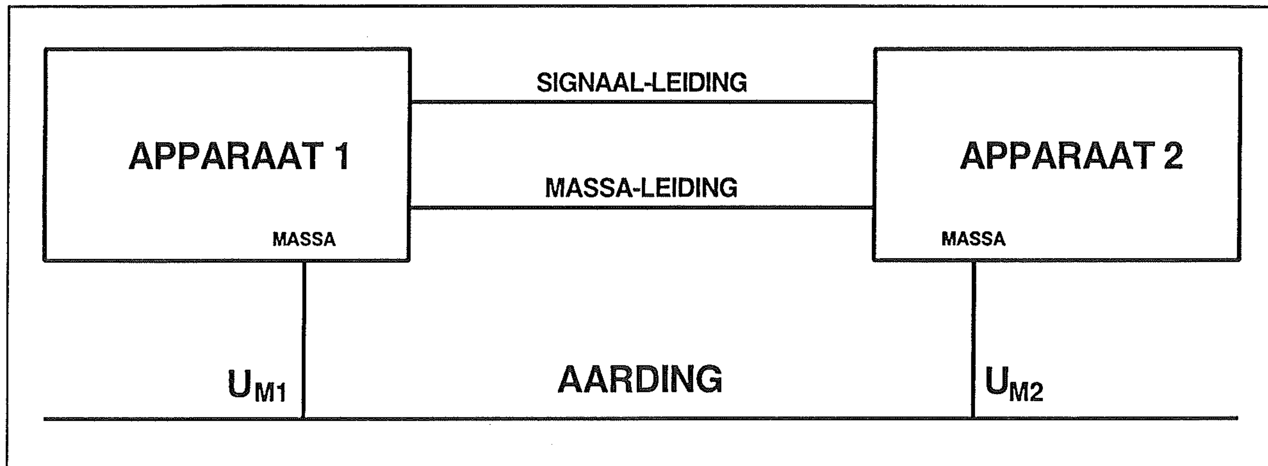
Massa-lussen

In figuur 4/6.6-1 is een reële situatie geschetst. Apparaat 1, bijvoorbeeld een elektronische thermometer ergens in een fabriek, zet de gemeten temperatuur om in een elektrische spanning en nadien in een pulstrein. Deze pulsen worden via een RS232C kabel verzonden naar apparaat 2, bijvoorbeeld de seriële poort van een computer in een centrale regelruimte. In de kabel zit natuurlijk een massa-leiding, die noodzakelijk is voor het sluiten van de stroomkring. Op deze manier kan de pulstrein van apparaat 1 naar apparaat 2 overgebracht worden. Apparaat 1 is natuurlijk met het net verbonden en negen kansen op de tien via een geaarde leiding. De massa van het apparaat ligt dus aan de aarde. Ook apparaat 2 zal via een geaarde leiding met het net verbonden zijn. Ook de massa van dit apparaat ligt dus aan de aarde. Nu is de kans groot dat de twee apparaten op heel verschillende netten staan. Weliswaar zijn de aardingscontacten van deze netten ongetwijfeld met elkaar verbonden, maar via welke wegen die

LEES OOK:
Hoofdstuk 6/6.11.1
Hoofdstuk 6/10.11



6.6 Optisch geïsoleerde RS232 interface



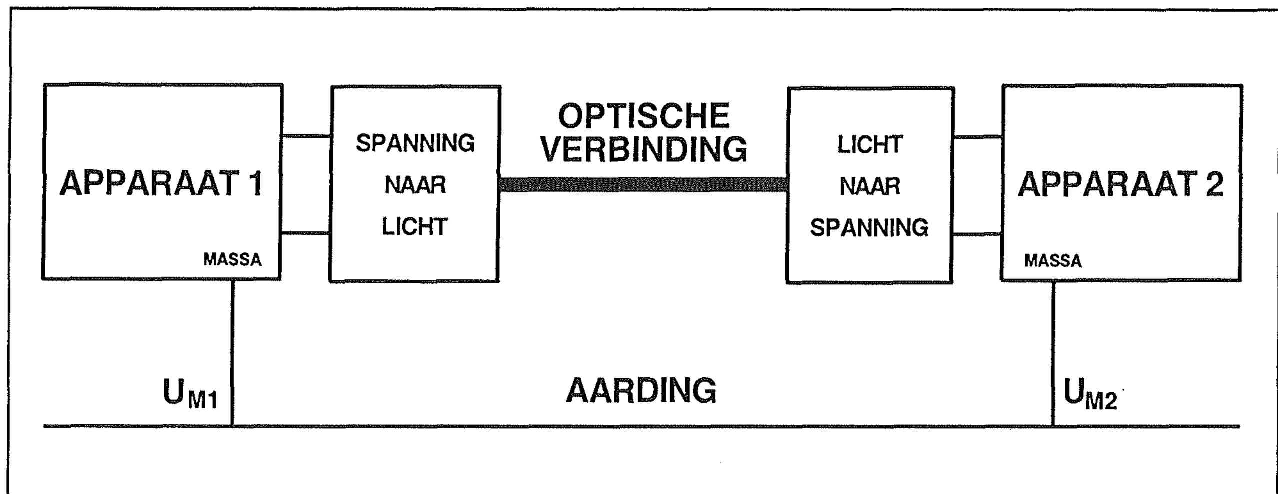
Figuur 4/6.6-1: Het ontstaan van massa-lussen.

verbinding tot stand komt is moeilijk uit te zocken.
Het kan best zijn dat de aardingsverbinding tussen beide netten kilometers lang is! Het zou dus helemaal niet zo vreemd zij als men vaststelt dat het aardingspotentiaal U_{M1} niet gelijk is aan het aardingspotentiaal U_{M2}. Dat betekent dat de massa's van beide apparaten niet op dezelfde spanning staan. Door de massa-leiding in de seriële kabel worden deze twee massa's echter met elkaar verbonden. Bovendien liggen de massa's ook nog eens aan elkaar via het aardingsnet. Er ontstaat dus een gesloten stroomkring, die men een ***massa-lus*** noemt. Als U_{M1} niet gelijk is aan U_{M2} gaat door deze lus een bepaalde aardingsstroom vloeien. Deze stroom vloeit dus door de massa-leiding van de seriële kabel tussen beide apparaten. In wezen is dit nog niet het grootste probleem. Maar tussen de massa-leiding en de signaal-leiding zit natuurlijk een vrij lage impedantie, zodat het kan gebeuren dat een deel van de massa-stroom niet door de massa-leiding vloeit, maar door de signaal-leiding! Dergelijke verschijnselen zijn moeilijk op te sporen en kunnen de goede data-overdracht volledig in de war schoppen.

Paracitaire capaciteiten
Een tweede probleem wordt gevormd door de paracitaire capaciteiten van de kabel. De signaal- en massa-leidingen lopen parallel door de kabel op een tamelijk kleine onderlinge afstand. Een dergelijk systeem vormt natuurlijk een capaciteit, waarvan de waarde afhankelijk is van de lengte van de kabel. Bij lange kabels moet men rekening houden met capaciteiten van honderden pF, die natuurlijk een zware belasting vormen voor de kabeldrivers in apparaat 1 en 2. Deze capaciteiten vormen een laagdoorlaat filtertje, dat de hoogfrequente harmonischen uit de datasignalen haalt. Een te zware capacitieve belasting zorgt ervoor dat de snelheid van betrouwbare data-transport dramatisch verlaagd wordt. Staan apparaat 1 en apparaat 2 op een grote afstand, dan zijn de paracitaire capaciteiten zelfs zo groot, dat data-transport niet eens meer mogelijk is.

Elektromagnetische straling
Hoewel RS232C kabels volledig afgeschermd zijn, zijn zij toch niet ongevoelig voor instraling door grote externe elektromagnetische velden. Deze stralingsbronnen kunnen stoorspanningen in de signaal-leidingen introduceren.

6.6 Optisch geïsoleerde RS232 interface

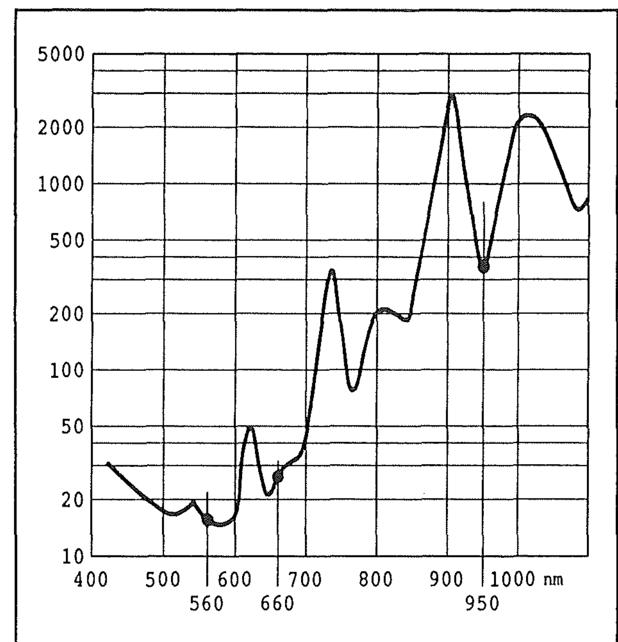


Figuur 4/6.6-2: Een optische verbinding tussen beide apparaten is dé oplossing voor alle problemen.

Deze kunnen de betrouwbaarheid van de data-overdracht beïnvloeden.

De oplossing

De oplossing voor dit soort problemen is in wezen heel eenvoudig en blokschematisch voorgesteld in figuur 4/6.6-2. Tussen apparaat 1 en apparaat 2 wordt een ***optische*** verbinding tot stand gebracht. Natuurlijk heeft men hiervoor een spanning naar licht omvormer en een licht naar spanning omvormer nodig, maar deze apparaatjes kunnen tegenwoordig heel snel en heel goedkoop gemaakt worden, zodat dit niet het probleem kan zijn. Het probleem is de prijs van de glasvezelkabel. Gelukkig bestaan er tegenwoordig vrij goedkope kabels met kunststofkern, die echter als nadeel hebben dat zij het licht fors dempen. Die dempingsfactor is bovendien sterk afhankelijk van de golflengte van het licht dat door de kabel wordt gestuurd. Uit de grafiek van figuur 4/6.6-3 volgt dat de demping stijgt met toenemende golflengte van het licht. Voor groen licht, met een golflengte van ongeveer 560 nm, ligt de kabeldemping het laagst, namelijk ongeveer 18 dB/100 m. Voor infrarood licht, met een golflengte van 950 nm, stijgt de demping tot niet minder dan 350 dB/100 m!



Figuur 4/6.6-3: De demping van een kunststof kabel in functie van de golflengte.

Hoewel het gebruik van groen licht dus voor de hand ligt, stuit men op het praktische bezwaar dat de meeste geïntegreerde licht-modulatoren en -demodulatoren

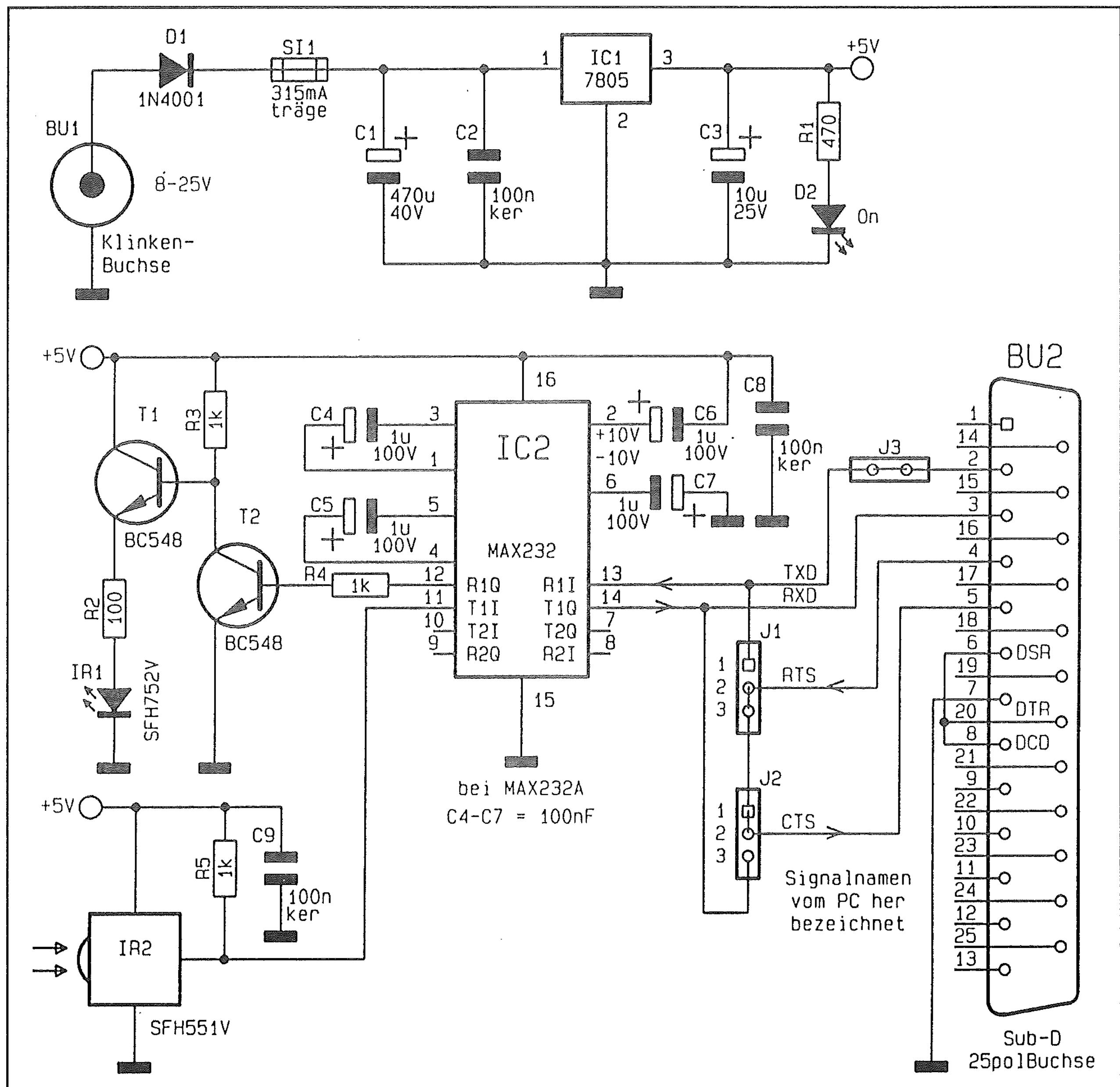
met infrarood werken. De hoge kabeldemping wordt echter voor een deel gecompenseerd doordat infrarode LED's een veel groter optisch vermogen hebben.

Eigenschappen van de schakeling
De in dit hoofdstuk voorgestelde schakeling zorgt voor de bidirectionele omzetting van RS232 signalen in lichtpulsen en van lichtpulsen in RS232 signalen. Wil men een optische RS232 link maken, dan heeft men twee identieke schakelingetjes nodig. De te overbruggen afstand is uiteraard afhankelijk van het soort lichtgeleidende kabel dat wordt toegepast, maar zelfs met de goedkope kabel met kunststof kern kan men zonder enig probleem afstanden van meer dan 50 m overbruggen en dit met een data-snelheid van maximaal 116 kBaud. De richtprijs van een dergelijke kabel bedraagt ongeveer *f* 2,50 per meter.
Als men werkt met bi-directionele data-overdracht, moeten beide schakelingen door middel van twee glasvezelkabels met elkaar verbonden worden.

Het schema
Het volledig schema van de optisch geïsoleerde RS232 interface is getekend in figuur 4/6.6-4. De verbinding tussen PC en de eerste interface en tussen het externe apparaat en de tweede interface komt tot stand met standaard RS232C kabels, die aangesloten worden op de standaard 25-polige Sub-D connector BU2.
De schakeling bezit drie jumpers J1, J2 en J3, waarmee men de onderstaande protocollen kan instellen:
- **Bidirectionele data-overdracht:**
  - Zender:
    J1: 2 en 3 verbinden
    J2: 1 en 2 verbinden
    J3: sluiten
  - Ontvanger:
    J1: 2 en 3 verbinden
    J2: 1 en 2 verbinden
    J3: sluiten
- **Unidirectionele data-overdracht:**
  - Zender:
    J1: 2 en 3 verbinden
    J2: 2 en 3 verbinden
    J3: sluiten
  - Ontvanger:
    J1: 1 en 2 verbinden
    J2: 1 en 2 verbinden
    J3: openen

Er wordt géén hardware-handshake toegepast, alleen de signalen TXD en RXD worden verzonden.
Bij het beschrijven van de werking van de schakeling wordt uitgegaan van bidirectionele data-overdracht, dus de pennen 4 en 5 van BU2 worden via de jumpers J1 en J2 met elkaar verbonden, en de jumper J3 wordt gesloten verondersteld. Verder zijn de RS-232 signalen DSR (pen 6), DCD (pen 8) en DTR (pen 20) met elkaar verbonden.
De controle over het protocol moet dus softwarematig gebeuren. In eerste instantie wordt het signaalverloop van de RS232 connector naar de optische zender IR1 beschreven. De data-stroom wordt ontvangen op pen 2 van BU2 en gaat via de jumper J3 naar een van de ingangen van de niveau-omzetter IC2. Deze zet de RS232 impulsen, met hun typische symmetrische schakelnivaus, om in TTL-compatibele signalen. Deze staan ter beschikking op pen 12 van dit IC.
De maximale snelheid van het systeem is in grote mate afhankelijk van de bandbreedte van de toegepaste niveau-omzetter.
De MAX232 heeft een gegarandeerde datastroom van 20 kb/s.

6.6 Optisch geïsoleerde RS232 interface



Figuur 4/6.6-4: Het volledig schema van de optische RS232 interface.

Er bestaat echter een "opgewaardeerde" versie, met als codering MAX232A, met een gegarandeerde bandbreedte van 116 kBaud. Welk IC men wil gebruiken is dus alleen afhankelijk van de maximale snelheid van het data-transport. Wel moet men, bij gebruik van het A-type, de vier condensatoren C4 tot en met C7 verlagen tot 100 nF. Op de printplaat kunnen zowel staande elco's (1 µF) als MKH's (100 nF) ingesoldeerd worden.

De TTL-compatibele signalen op pen 12 worden aangeboden aan de basis van transistor T2. Deze trap zorgt voor een invertering van de data-stroom. De collector van deze schakeltrap is rechtstreeks verbonden met de basis van de emittervolger T1. De zender-module IR1 (SFH752V

6.6 Optisch geïsoleerde RS232 interface

van Siemens) staat in de emitter met een stroombegrenzende weerstand R2 in serie.

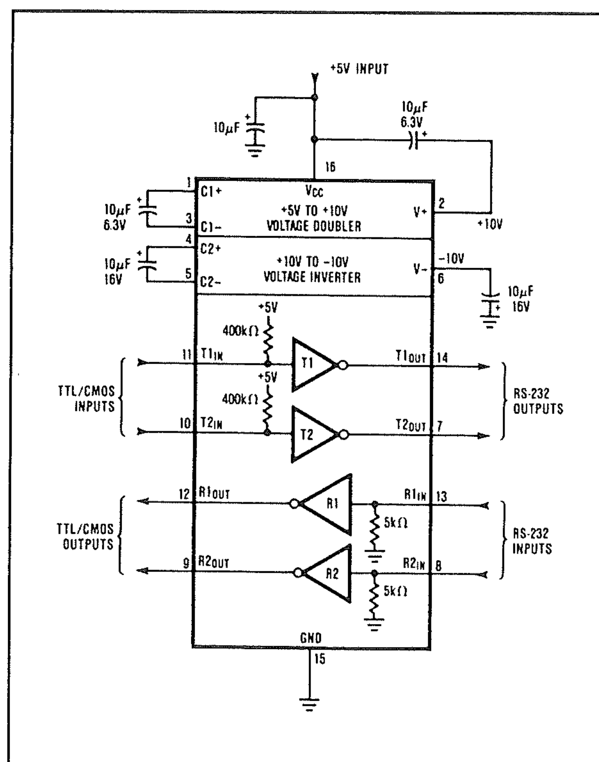
Voor het ontvangen van de optische gegevens wordt gebruik gemaakt van een ontvanger-module van dezelfde fabrikant, met als typenummer SFH551V (IR2).

Dit module bestaat uit een zeer gevoelige en snelle infrarode diode en een snelle versterker die TTL-compatibele uitgangssignalen levert. Deze versterker heeft echter een open-collector uitgang, zodat de belastingsweerstand R5 extern moet worden aangebracht. Het uitgangssignaal van het module gaat naar pen 11 van IC2. Het IC voert nu een omgekeerde niveauomzetting uit, de TTL-gegevens worden omgezet in RS232-compatibele signalen op pen 14. Deze worden rechtstreeks uitgevoerd naar pen 3 van de RS232-connector BU2.

De MAX232 kan werken met een enkelvoudige voedingsspanning van +5 V. Zoals uit het interne blokschema blijkt, getekend in figuur 4/6.6-5, bevat de schakeling twee geïntegreerde DC/DC-omzetters, die uit de +5 V symmetrische spanningen van +10 V en -10 V fabriceren. Hiervoor worden de bekende pompschakelingen gebruikt, die bestaan uit analoge schakelaars die twee condensatoren voortdurend opladen tot de voedingsspanning. Andere schakelaars zorgen er voor dat de twee condensatoren op de juiste manier in serie worden geschakeld, waardoor de voedingsspanning verdubbeld wordt. De gestabiliseerde +5 V wordt opgewekt uit een netstekervoeding van minimaal 8 V tot maximaal 25 V en gestabiliseerd met behulp van IC1.

Bouw van de schakeling

De volledige schakeling kan ondergebracht worden op het kleine printje, getekend in figuur 4/6.6-6. De componentenopstelling is getekend in figuur 4/6.6-7. De bouw van de schakeling behoeft voor de ervaren H.E.-lezer(es) natuurlijk geen enkele toelichting, zeker niet na extra bestudering van het uiterlijk van het prototype, voorgesteld in figuur 4/6.6-8. De spanningsstabilisator IC1 wordt plat op de print gemonteerd en met behulp van een M3x10 boutje en een M3 moertje vastgeschroefd.



Figuur 4/6.6-5: Het intern blokschema van de MAX232.

Werken met glasvezelkabel

Een verbinding met glasvezelkabel maken is uiteraard iets ingewikkelder dan koperen adertjes aan de contacten van een connector solderen. Bij het doorknippen van een glasvezelkabel ontstaan (op microscopische schaal) zeer groffe snijvlakken, die voor optische demping zorgen en het licht in alle richtingen verstrooien.

6.6 Optisch geïsoleerde RS232 interface

ONDERDELENLIJST

WEERSTANDEN, 1/4 W, 5 %	
R1	470 Ω
R2	100 Ω
R3,R4,R5	1 kΩ
CONDENSATOREN	
C1	470 µF 25 V printelco
C2,C8	100 nF ceramisch
C3	10 µF 16 V printelco
C4,C5,C6,C7	1 µF 16 V printelco
C4',C5',C6',C7'	100 nF MKH
HALFGELEIDERS	
D1	1N4001
D2	LED, 5 mm, rood
T1,T2	BC548
IC1	7805
IC2	MAX232(A)
IR1	SFH752V
IR2	SFH551V
DIVERSEN	
1 x	25-polige Sub-D connector, print
1 x	16-polig IC-voetje
1 x	printjumper, 2 contacten
2 x	printjumper, 3 contacten
1 x	printzekeringhouder
1 x	zekering 315 mA, traag
1 x	3,5 mm connector, print
1 x	M3x10 boutje
1 x	M3 moertje

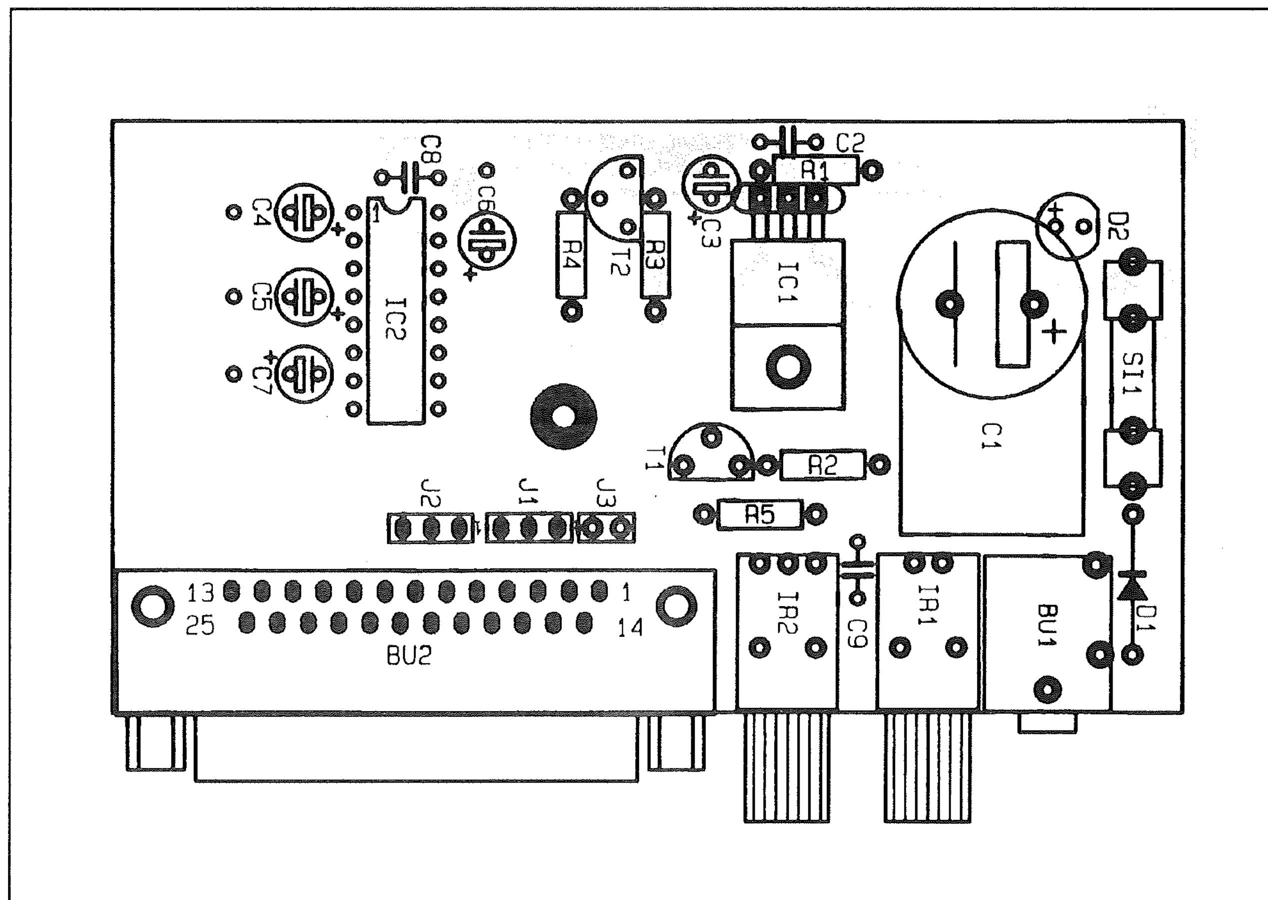
Gelukkig is kunststof glasvezelkabel vrij gemakkelijk te bewerken. Zowel de zender IR1 als de ontvanger IR2 zijn voorzien van zelf-centrerende schroefbevestigingen, waarin men het uiteinde van de glasvezelkabel kan klemmen, zie figuur 4/6.6-9. Het snijden van de kabel kan zonder de speciale instrumenten, die wél noodzakelijk zijn als men "echte" zeer dunne monomodus glasvezel zou toepassen. Voor het overbruggen van korte afstanden (10 m) volstaat het de kabel met een scherp mes loodrecht op de lengterichting door te snijden en deze uiteinden nadien in de SFH-componenten te monteren. Om de demping te reduceren wordt aanbevolen de snijvlakken te bewerken met polijstpapier met grofheid 600.
Als men langere afstanden moet overbruggen gaat men als volgt te werk. Een zeer scherp mesje wordt verhit tot een temperatuur van ongeveer 170 °C. Hiermee snijdt men de kabel ***loodrecht*** door. Nadien wordt het afgesneden uiteinde van de kabel ***loodrecht*** vijf seconde lang tegen een tot 120 °C verhitte metalen plaat geduwd.
Bij het monteren van de kunststof-kern lichtgeleidende kabel moet men er rekening mee houden dat scherpe hoeken

absoluut uit den boze zijn. De minimale buigingsstraal bedraagt 20 mm.

Bouwpakket informatie

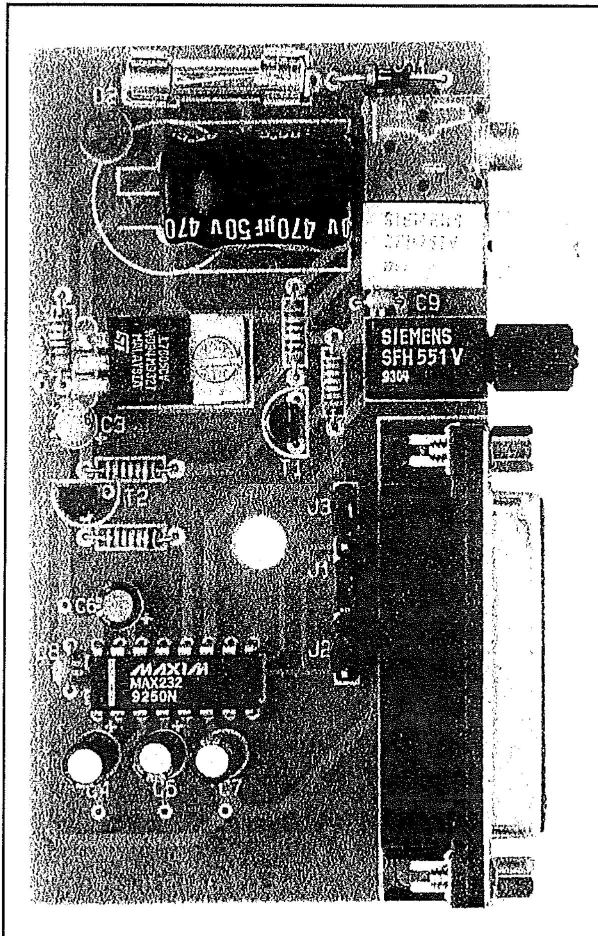
Als extra service aan de nabouwers van deze ELV-schakeling kan nog vermeld worden dat deze optisch geïsoleerde RS232 interface in diverse onderdelenzaken leverbaar is als compleet bouwpakket.

De samenstelling van dit bouwpakket, inclusief de print en een voorgeboord en bedrukt kastje, wordt verzorgd door de firma DIL, Postbus 5544, 3008 AM Rotterdam, telefoon 010-485.42.13. Het bouwpakket wordt geleverd onder de bestelcode 40-155-79. Op het genoemde adres kan men alle nodige informatie krijgen over prijzen en verkoopadressen.

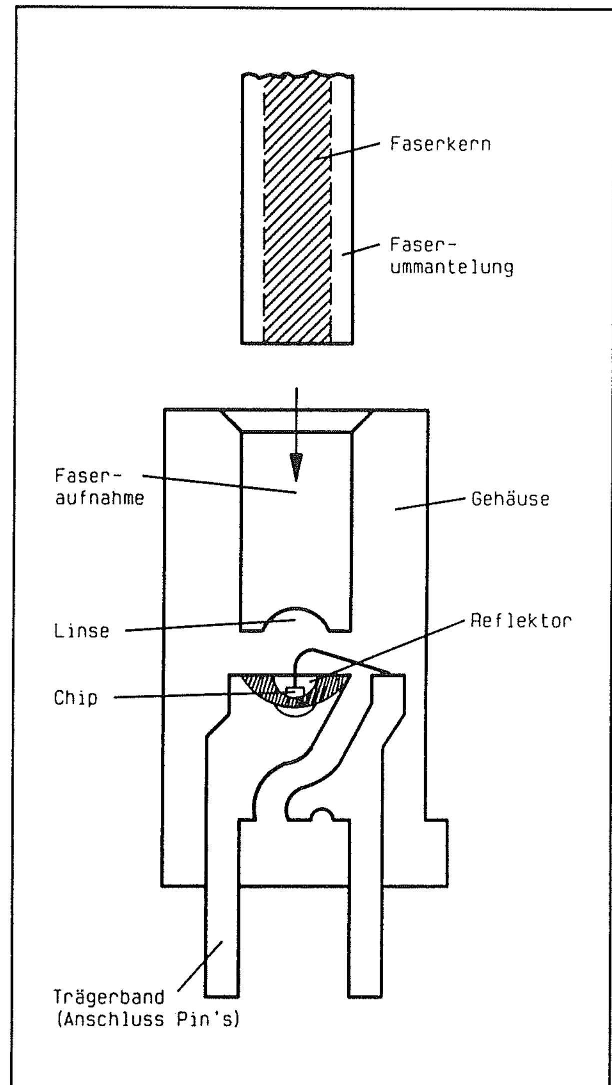


Figuur 4/6.6-7: De componentenopstelling van de schakeling.

6.6 Optisch geïsoleerde RS232 interface



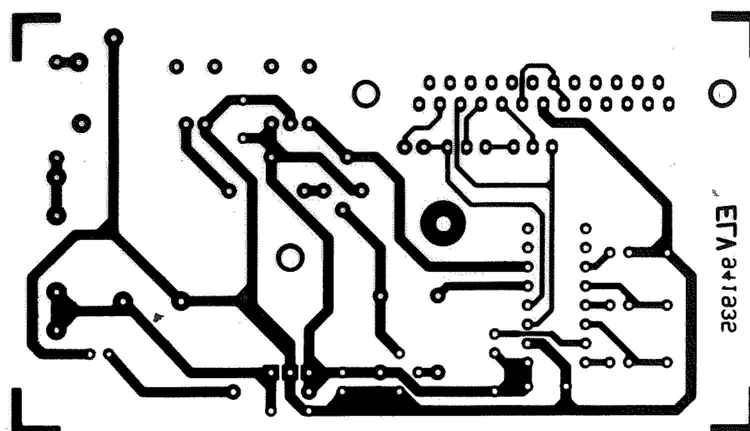
Figuur 4/6.6-8: Het proto-type van de schakeling.



Figuur 4/6.6-9: De montage van de glasvezelkabel aan de Siemens-modulen.

6.6 Optisch geïsoleerde RS232 interface

6.6 Optisch geïsoleerde RS232 interface



Figuur 4/6.6-6: De print voor de schakeling.

4/6.7.1

De bouw van Chip

Inleiding

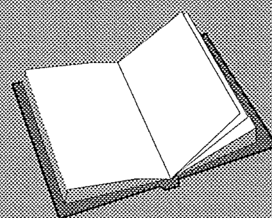
Twee lucifersdoosjes
Chip is een computertje ongeveer zo groot als twee luciferdoosjes tegen elkaar. Het hart van Chip is een microcontroller. Heel bijzonder is, dat in de microcontroller geen programma zit voor een vaste taak, maar een programma dat instructies in een hogere programmeertaal decodeert en ze in microcontroller code uitvoert.
Deze programmeertaal is speciaal geschreven voor Chip en heel eenvoudig van opzet. Om met Chip te werken is kennis van microcontrollers niet nodig. Het enige dat nodig is, is Chip's instructieset.

Standaard hardware
Chip bevat standaard alle hardware die voor eenvoudige toepassingen nodig is en voor de sturing daarvan bevat de hogere programmeertaal instructies. Zo bevat Chip analoge en digitale in- en uitgangen, aansluitingen voor een LCD-display en een toetsenbord. Kortom, Chip kan volwaardig met de buitenwereld communiceren.

Programmeren via de PC
Om Chip te gebruiken is alleen een PC nodig met een seriële poort. Chip bevat alle software die voor de communicatie nodig is. Het programmeren van Chip kan direct in de hogere programmeertaal, aan de hand van de instructieset, maar ook in de symbolische schrijfwijze met **mnemonics** en **labels**. Door een **assembler** worden de symbolische programma's in instructies vertaald en kunnen dan meteen in Chip worden geladen.
Deze manier van programmeren levert heel leesbare programma's en maakt de kans op fouten kleiner.

Krachtig computersysteem
Ondanks alle eenvoud is Chip een vrij krachtig computersysteem. Maar om alle mogelijkheden vol te benutten is enige kennis van elementaire digitale principes onmisbaar. Ook begrippen als bytes, nibbles en bits moeten vertrouwd klin-

LEES OOK:
Hoofdstuk 6/10.6



ken en de elementaire werking van het computermodel volgens von Neumann moet bekend zijn.
Als aan deze voorwaarden wordt voldaan, zal men van Chip veel plezier kunnen hebben en het kleine apparaatje steeds meer gaan waarderen.

Kennismaking met Chip
Chip is een computertje dat speciaal is ontworpen om kleine projecten te automatiseren. Dat kan een woning zijn, een alarminstallatie, een robot, een datalogger, een weerstation, kortom systemen waarin de tijd een rol kan spelen en waar signalen moeten worden gemeten en relais en/of servo's aangestuurd.
Het hart van Chip is een microcontroller, die een operating system bevat voor de communicatie met de gebruiker én een interpreter voor het uitvoeren van programma's, die in een interne EEPROM zijn opgeslagen. Zo'n programma bestaat uit een reeks macro-instructie's, elk twee bytes lang, die in machinetaal geschreven routines in de microcontroller oproepen en sturen.
Een Chip programma kan gewoon in "mensentaal" worden geschreven. Een assembler stelt de macro's samen, die dan met een uploader in de EEPROM worden geschreven.
Om met Chip te kunnen werken is een PC nodig met een seriële poort, dat is alles.
Een aardige toepassing is die van een robot, en dat zullen we in een volgend hoofdstuk verder uitwerken. Chip is ook heel geschikt om woningen te automatiseren. Zo kan de verwarming worden geregeld aan de hand van de binnen- en buitentemperaturen, de dag van de week en de tijd. Op een display worden de gegevens zichtbaar gemaakt en met een toetsenbordje kunnen bedieningsfuncties worden opgeroepen.
Als u de Basic Stamp [1] kent, dan is het u misschien opgevallen dat tot zover Chip wel enige gelijkenis vertoont met de Basic Stamp zélf. Toch is dat zuiver toeval, want het concept van de in Chip gebruikte macrotaal is niet nieuw [2].

Specificaties
De eigenschappen van Chip kunnen als volgt worden samengevat:
- kenmerk:
  programmeerbare minicomputer op basis van een ST62T65 microcontroller
- programmeertaal:
  macro-instructies van 2 bytes
- type vertaler:
  interpreter
- aantal variabelen:
  16, 0...F
- instructietijd:
  ongeveer 1 ms, afhankelijk van type instructie
- programmageheugen:
  I^2C EEPROM met 2.048 bytes
- ingangen:
  vijf, digitaal en analoog leesbaar
- keyboard:
  twaalf key keyboard mogelijk op analoge ingang
- uitgangen:
  - vijf, digitaal met 20 mA sink/7 mA source (begrensd door weerstanden)
  - twee servo's, pulstijd 0,9 ms - 1,9 ms, frequentie 50 Hz
  - alternatief voor servo-uitgangen is één analoge uitgang
  - LCD-display, 1 regel 16 karakters
  - sounder
- lopende tekst:
  maximaal 24 karakters op het LCD

- real-time klok:
  weken, dagen, uren, minuten en seconden
- klok fine-tuning:
  softwarematig in stappen van 2,7 ppm
- timers:
  korte timer, seconden en minuten timers en een sounder
- communicatie:
  interactieve commandoprocessor, RS232, 19200,7,n,2
- voedingsspanning:
  4,4 V - 6,0 V
- voedingsbron:
  accupakket van 4 NiCad's of 5 V voeding
- stroomverbruik:
  ongeveer 3 mA zonder externe componenten
- print afmetingen:
  70 x 62 mm
- hulp software:
  Chip assembler, Chip terminal met hex uploader
- toepassingen:
  woning-automatisering, robots, lange tijd logger, eenvoudige besturingen, alarminstallaties, technische modelbouw

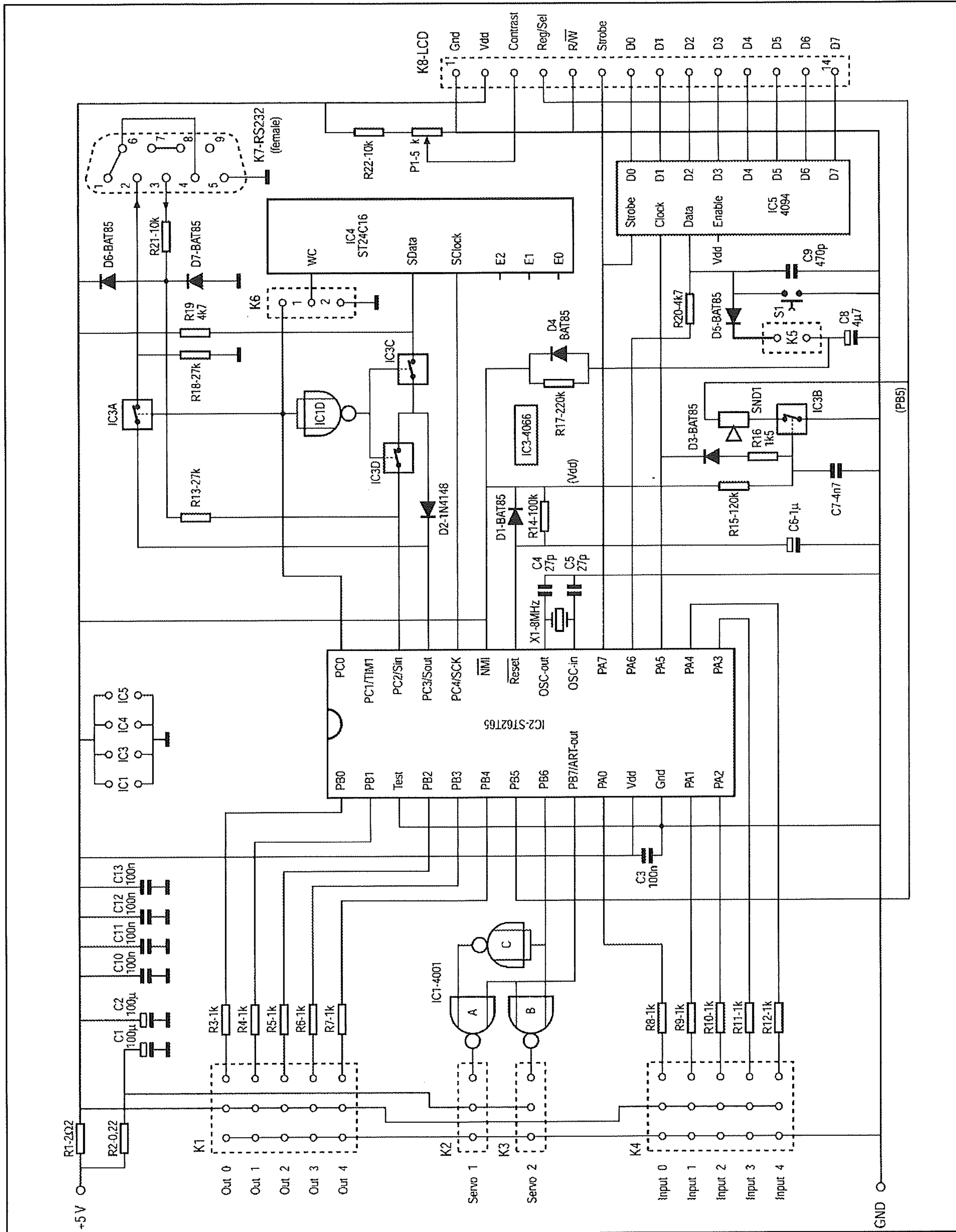
Systeembeschrijving
Chip bevat niet alleen een geprogrammeerde microcontroller, maar ook een aantal hardware componenten om hem zo veelzijdig mogelijk te maken. De hardware kan direct met macro instructies worden gestuurd. Het is dus niet nodig om te weten hoe de microcontroller zélf werkt, alles kan in Chip code worden bestuurd. Toch kunnen we ons voorstellen, dat u meer over de gebruikte controller wilt weten. In [3] is een boek vermeld, dat veel aspecten op een duidelijke manier uitlegt. Ook van het Internet zijn de datasheets te downloaden (www.st.com). Voor de echte specialisten is het misschien wel interessant om te weten, dat het mogelijk is om door middel van Chip-code de registers in de microcontroller te lezen en te schrijven.

Snel programmeren
Door de efficiënte macro-instructies zijn Chip programma's heel compact en snel te schrijven. Het assembleren van een programma is in seconden gebeurd, waarna het onmiddellijk geladen en getest kan worden. Het aanbrengen van wijzigingen is vrijwel interactief, wat het werken heel plezierig maakt.

De hardware

Het schema
In figuur 4/6.7.1-1 is het volledig schema van Chip voorgesteld. Centraal daarin staat de microcontroller ST62T65 van ST-Microelectronics (IC2). Het is een CMOS IC zoals ook de andere IC's, om het stroomverbruik laag te houden. De microcontroller bevat een aantal extra componenten, zoals een gewone timer, een auto-reload timer, een analoog/digitaal-converter en een seriële perifere interface, kortweg "SPI". Alle perifere componenten worden door Chip gebruikt. De gewone timer wordt gebruikt als klok en houdt de seconden, minuten, uren, dagen en weken van het jaar bij. Ook de diverse software timers worden door de gewone timer bestuurd. De auto-reload timer wordt gebruikt om twee servo's onafhankelijk van elkaar te sturen. Beide timers werken op interruptbasis, waardoor Chip in feite "multitasking" is. De afregeling van de klok gebeurt met een software byte, dat door de gebruiker kan worden ingesteld.

6.7 Chip, een zelfbouw computertje



Figuur 4/6.7.1-1: Het volledig schema van Chip.

6.7 Chip, een zelfbouw computertje

Eenmaal per minuut wordt de timer met deze waarde gecorrigeerd.
Op K1 zijn vijf digitale uitgangen beschikbaar (Out 0 - Out 4) met serieweerstanden om de microcontroller te beveiligen. Deze uitgangen kunnen worden gezet en teruggezet en de toestand van iedere uitgang kan worden gelezen. Op de servo-uitgangen K2 en K3 kunnen normale servo's worden aangesloten, zoals die voor radiografisch bestuurde modellen in soorten en maten verkrijgbaar zijn. De servo's worden onafhankelijk van elkaar door de auto-reload timer gestuurd. Een interruptroutine schakelt via PB6 poort A of poort B van IC1 door.
Op connectorblok K4 zijn vijf ingangen aanwezig. Elke ingang kan digitaal en analoog worden gelezen. In het laatste geval wordt de gekozen ingang met de ADC verbonden. Bij iedere in- en uitgang zijn op de connectors "Gnd" en "V_{dd}" aansluitingen aanwezig.
Sensors, LED's en dergelijke kunnen direct op de connectors worden aangesloten. Servo's met Futaba of JR servostekers passen direct op de servo aansluitingen. Omdat belaste servo's aanzienlijke stromen opnemen, worden ze via een separaat ontkoppelde verbinding vanuit het voedingspunt van de schakeling gevoed.
Voor de communicatie met de gebruiker dient connector K7. Dit is een standaard sub-D connector, die met een verlengsnoer op een seriële poort van de PC wordt aangesloten. Het ingangssignaal wordt door R21, D6 en D7 begrensd en via R13 naar de ingang van de SPI (Sin) gevoerd. Het seriële uitgangssignaal (Sout) bereikt via analoge schakelaar IC3A de uitgangspen van de connector. Als de analoge schakelaar open is, wordt door R18 de uitgang laag gehouden. De niveau's van het seriële uitgangssignaal zijn niet conform de RS232 norm, maar toch werkt deze schakeling betrouwbaar mits het verbindingssnoer niet te lang is.
Bij seriële communicatie is PC0 hoog en IC3A gesloten, IC3C en IC3D zijn geopend. Door PC0 laag te maken, wordt de seriële EEPROM (IC4) met de SPI verbonden en via WC (Write Control) schrijfbaar gemaakt. Nu kunnen de softwareroutines via het I^2C-protocol [4] de EEPROM voor lezen en schrijven benaderen. Voor de feitelijke overdracht zorgt de SPI met een klokfrequentie van 308 kHz.
Op connector K8 kan een éénregelig karakter-LCD met zestien letters worden aangesloten. Het is geen écht 16 karakter LCD, maar een 2 x 8 karakter LCD, het meest voorkomende type. Over LCD's kan een heel boek worden geschreven, in [5] staan enkele lezenswaardige publicaties.
Voor de aansturing van de databus van het LCD wordt een serieel naar parallel omzetter gebruikt (IC5) zodat voor de overdracht slechts drie uitgangen van de microcontroller nodig zijn. PA5 levert de klok, PA6 de data en PA7 de strobe. De strobe van IC5 is positief, zodat PA7 tevens kan worden gebruikt als (negatieve) strobe om de data in het LCD te klokken. Normaal is PA6 als ingang geschakeld. Op deze ingang is drukknop S1 aangesloten. Deze wordt eenmaal per seconde door het operating system getest en als de drukknop is ingedrukt wordt een draaiend Chip programma gestopt en teruggesprongen naar de commandoprocessor. S1 is dus geen resetknop maar een breakknop en de realtime klok blijft doorlopen. Aansluiting Reg/Sel van het LCD dient voor de keuze karaktermode of commandomode.

6.7 Chip, een zelfbouw computertje

ONDERDELENLIJST

WEERSTANDEN, 1/4 W, 5 %

R1	2,2 Ω
R2	0,22 Ω
R3-R12	1 kΩ
R13,R18	27 kΩ
R14	100 kΩ
R15	120 kΩ
R16	1,5 kΩ
R17	220 kΩ
R19,R20	4,7 kΩ
R21,R22	10 kΩ

INSTELPOTENTIOMETER, PIHER, STAAND, 6 mm

P1	5 kΩ

CONDENSATOREN, RM 2,54

C1,C2	100 µF	16 V printelco
C3,C10-C13	100 nF	multilayer
C4,C5	27 pF	ceramisch
C6	1 µF	35 V tantaal
C7	4,7 nF	ceramisch
C8	4,7 µF	35 V tantaal
C9	470 pF	ceramisch

HALFGELEIDERS

IC1	4001
IC2	ST62T65, geprogrammeerd
IC3	4066
IC4	ST24C16
IC5	4094
D1,D3-D7	BAT85
D2	1N4148

DIVERSEN

X1	kristal 8 MHz HC 49/S
SND1	passieve buzzer ø14 x 7,5 RM 7,5 (Elproma type EPM121A1AWP-T of Farnell best. nr. 926-899)
K1,K4	5 x 3-pin male printheader
K2,K3,K6	3-pins SIL male printheader
K5	2-pins SIL male printheader
K7	9-polige haakse sub-D connector, female, voor printmontage
K8	14-polige SIL printheader male
1	IC-voet 8 pens, buscontacten (PreciDip)
2	IC-voeten 14 pens, buscontacten (PreciDip)
1	IC-voet 16 pens, buscontacten (PreciDip)
1	IC-voet 28 pens, buscontacten (PreciDip)
S1	drukknop type D6 (ITT/Schadow)
17	draadbruggen
1	print Chip
LCD	display 1 regel x 16 karakters Display best. nr. 71.52.1116, Conrad best. nr. 183261, Farnell best. nr. 142-542)

6.7 Chip, een zelfbouw computertje

Deze aansluiting wordt door PB5 gestuurd. Tevens wordt PB5 voor de sturing van sounder SND1 gebruikt. Bij de eerste negatief gaande klok, gebruikt om data voor het LCD in het schuifregister te zetten, wordt analoge schakelaar IC3B geopend en de sounder uitgeschakeld (D3, R16, C7). Na afloop van de data-transfer wordt C7 via R15 geladen en IC3B gesloten. Met P1 kan het contrast van het LCD worden ingesteld. Dit hoeft over het algemeen maar één keer te worden gedaan. Als voor het LCD een type met backlight wordt genomen is dat ook bij weinig of geen omgevingslicht goed afleesbaar.
De penbezetting van K8 komt overeen met die van de in de onderdelenlijst genoemde LCD's.
De power-up reset wordt verzorgd door D1, R14 en C6.

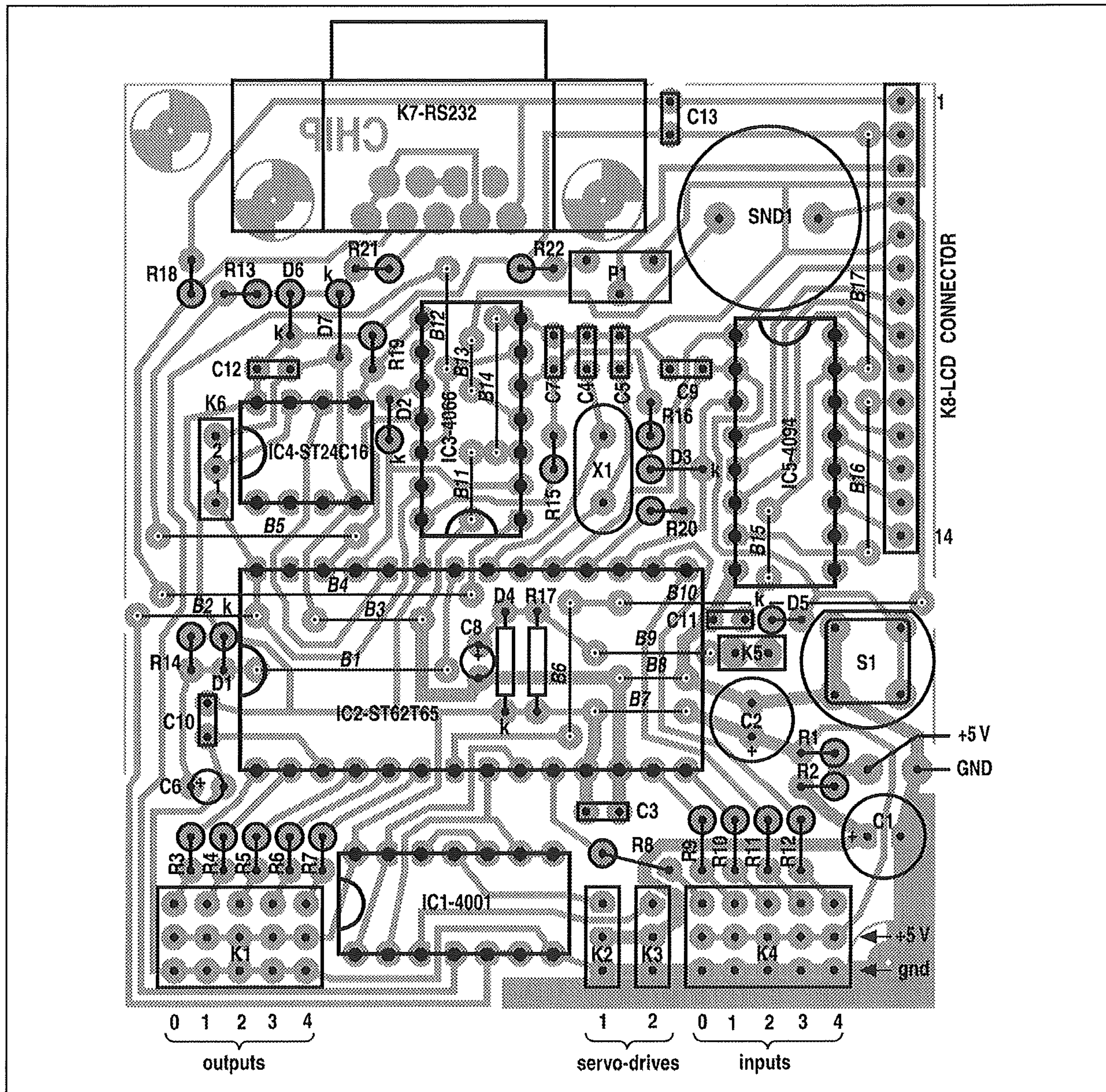
De bouw van Chip
In figuur 4/6.7.1-2, op de laatste pagina van dit hoofdstuk, is de **enkelzijdig print** voorgesteld op schaal 1:1 en in figuur 4/6.7.1-3 de opstelling van de onderdelen. Boor eerst alle gaten met een boortje van 0,8 mm. Ga nadien de volgende gaatjes uitboren:

- de gaten voor de connectoren en de sounder met 1 mm;
- de vier bevestigingsgaten met 3 mm.

Monteer dan eerst de 17 draadbruggen en breng over B10 ter hoogte van B15 een isolatiekousje aan zodat daar geen kortsluiting kan ontstaan. Gebruik bij voorkeur de aanbevolen voetjes, dan zijn er geen problemen met de draadbruggen. Let op de uitsparing in de voetjes ter indicatie van pen 1. Let op verder op dat, voor de voet voor IC2 wordt geplaatst, eerst C8, D4 en R17 worden gemonteerd.

Leg C9 plat op de print in verband met de hoogte én de brug in het IC-voetje. Voor connectoren K1 en K4 is onder een strip dubbelrijige pinheaders gebruikt en boven een strip enkelrijige, alle connectors behalve K7 zijn male. De connectoren zijn met een figuurzaagje voor metaal van de strip afgezaagd.
Als voedingsbron zijn vier penlight NiCad's in een batterijhouder gebruikt. In het snoer met batterijclip is een aan/uit schuifschakelaar (zelfreinigend) opgenomen.
Het snoer is direct in de voedingspunten van de print gesoldeerd. Door de batterijclip kan het accupakket worden losgenomen om te worden opgeladen.
In plaats van R2 hebben we een tweepolige male header gemonteerd. Uit een glaszekering van 1,2 A hebben we het zekeringdraadje gehaald en op een tweepolige female SIL-header gesoldeerd. Deze "zekering" is op de header op de print gezet.
Met een EEPROM ST24C16 kan op positie 1 van K6 een jumper worden gezet. Dan is de EEPROM normaal tegen schrijven beschermd. Jammer genoeg is bij sommige EEPROM's van ander fabrikaat de reactietijd van de Write Control niet snel genoeg. Dan kan WC niet actief worden gebruikt en wordt vast laag gemaakt door een jumper op positie 2 van K6.
Als de seriële ingang (K7, pen 3) laag is, dan staat op knooppunt R21, D6, D7 circa -0,3 V. Weerstand R13 vormt nu met de inwendige pull-up weerstand van Sin (nominaal 100 kΩ) een spanningsdeler. In het extreme geval dat deze pull-up weerstand een lagere waarde heeft dan ca. 60 kΩ, kan het nodig zijn om R13 iets te verkleinen, zodat het niveau op Sin als laag wordt herkend (max. 0,3 x V_{dd}).

6.7 Chip, een zelfbouw computertje



Figuur 4/6.7.1-3: De componentenopstelling van Chip.

Opmerking

Het is mogelijk om de Chip-interpreter zelfstartend te maken door over K5 een jumper te zetten. Door C8 en D5 wordt dan S1 even “ingedrukt gehouden” als Chip wordt aangezet. Door R17 wordt C8 verder opgeladen en wordt de drukknop vrijgegeven. Hetzelfde resultaat wordt bereikt, zonder de jumper over K7, door S1 tijdens het inschakelen ingedrukt te houden.

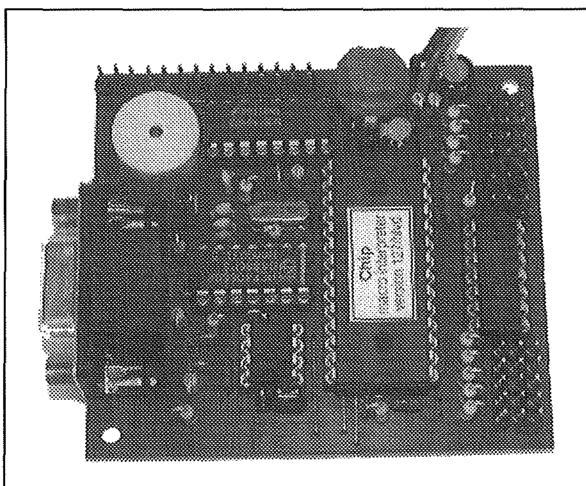
Let op

Van de ST62T65 zijn twee uitvoeringen leverbaar namelijk met achtervoegsels BB6 en met achtervoegsels CB6 of C. Bij de “B”-versie is de reset een ingang, maar bij de “C’-versie een uitgang. Bij de

"C"-versie moet daarom resetcondensator C6 worden weggelaten. De reset wordt door de controller zélf verzorgd door de LVD-optie (low voltage detector). Als u de microcontroller zelf programmeert, moet het LVD-bit van de Option Bytes worden gezet en de Option Bytes worden geprogrammeerd. Bij de "B"-versie hoeft de Option Byte niet persé te worden geprogrammeerd.

Het eindresultaat

Als de print volgens de handleiding is bestukt ziet het eindresultaat er uit zoals voorgesteld in figuur 4/6.7.1-4.



Figuur 4/6.7.1-4: Het prototype van Chip, dit wijkt iets af van de definitieve versie.

Het operating system

Inleiding

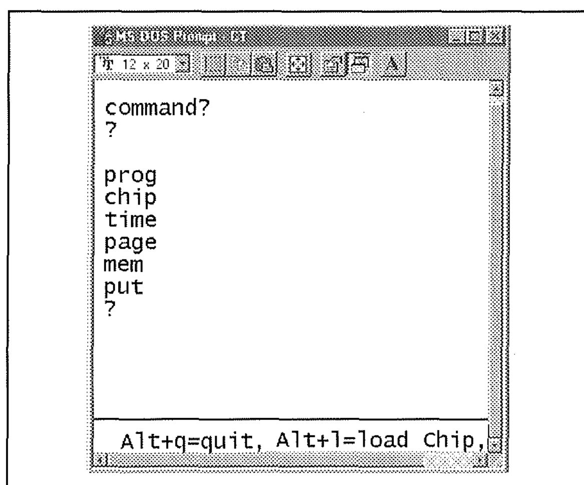
Chip bevat een eenvoudige commandoprocessor, die via de seriële poort met het terminal programma in de PC communiceert. Het protocol is 19200,7,n,2. In eerste instantie kan bijvoorbeeld het bij Windows geleverde Terminal programma worden gebruikt, maar om ook programma's naar Chip te uploaden moet **Chipterm.exe** of **Chip VB Terminal** worden gebruikt.

Downloaden

Beide programma's zijn te downloaden van www.hobbyelektronica.nu. Selecteer in het linker frame de optie "Software-service". U ziet in het rechter frame dan het aanvullingsnummer, de rest wijst zichzelf.

Commando's

Voor de ingave mogen uitsluitend kleine letters worden gebruikt. Na ingave van een commando en eventuele parameters, wordt afgesloten met "Enter" waarna de commandoprocessor in actie komt. Met "Backspace" kunnen fouten worden hersteld en met "Escape" wordt de hele regel weggegooid. Er zijn zeven commando's, zie figuur 4/6.7.1-5, we zullen ze in vogelvlucht behandelen want het wijst zich eigenlijk vanzelf.



Figuur 4/6.7.1-5: De beschikbare commandoset.

Als het Terminal Programma is gestart en Chip wordt aangezet, verschijnt na een druk op Enter: "command?" met op de volgende regel de prompt "?". Door

6.7 Chip, een zelfbouw computertje

het commando "?" te geven (afsluiten met Enter), wordt de commandoset op het scherm gezet.

prog [adres]
Met dit commando kunnen we in de externe EEPROM schrijven en lezen. Alleen de even adressen worden getoond, gevolgd door de bytes op dit adres en het erop volgende.
Het laagste adres is 0000 en het hoogste 07FF.
Met de "+" en "-" toetsen kunnen we het adres verhogen of verlagen. Na de ingave van 2 bytes worden deze weggeschreven en het adres verhoogd. Met Escape keren we terug naar de commandoprocessor.

chip
Dit commando start een Chip programma. De commandoprocessor werkt dan niet meer. Een Chip programma kan ook worden gestart door tijdens het aanzetten van Chip drukknop S1 ingedrukt te houden of op K5 een jumper te zetten. Het programma kan worden gestopt door op de drukknop te drukken, maar ook de break instructie of een fout in het programma doen het programma stoppen.

time
Na ingave verschijnt een lopende 24-uurs klok op het scherm. Drukken op Enter brengt ons in het weekveld, de klok kan worden gelijkgezet. Tweemaal drukken op Escape laat de klok ongewijzigd.

page, mem en put
Met deze commando's kunnen registers in de microcontroller worden gelezen en geschreven.

Adresbereik 00h-3Fh
Eerst moeten we echter iets vertellen over adresbereik 00h-3Fh. De microcontroller heeft behalve normale RAM ook twee EEPROM pagina's en een extra RAM pagina. Met het page commando kan een van deze drie pagina's in adresbereik 00h-3Fh worden geprojecteerd. In EEPROM pagina 0 zetten we van 00h-0Fh een introtekst en op adres 10h de afregelbyte van de klok.
Kies eerst met "mem 0" adres 00 van de microcontroller. Dit adres en de bijbehorende byte verschijnen op het scherm, het is een adres in de extra RAM pagina (page 2). Met "page 0" kiezen we EEPROM pagina 0. Met behulp van het commando put "byte" kan "byte" op dit adres worden geschreven. Met "+" wordt het adres verhoogd (met "-" verlaagd). Nu kan de welkom-boodschap in de EEPROM worden gezet:

2a, 20, 48, 69, 2c, 20, 49, 27, 6d, 20, 43, 68, 69, 70, 20, 2a

Als alle bytes in EEPROM pagina 0 staan, drukken we op Escape om naar de commandoprocessor terug te keren. Als u een lijst met de ASCII-karakters heeft, weet u waarschijnlijk al wat de introtekst is.
Om hem op het LCD te zetten moet Chip worden gereset door hem even uit en weer aan te zetten. Op adres 10 van dezelfde pagina zetten we op dezelfde manier de afregelbyte voor de klok. Met de waarde EAh zal hij al vrijwel gelijk lopen. Dus "mem 10", "page 0", "put ea". Verhogen van deze waarde doet de klok langzamer lopen, verlagen sneller. De invloed is pas na 24 uur of langer merkbaar. Vergeet niet om na iedere aanpassing Chip te resetten, want alleen bij het

opstarten wordt de byte uit de EEPROM gelezen, evenals de introtekst.

Referenties

[1] Elektuur, september 1999 - april 2000, Basic Stamp Cursus.

[2] Joseph Weisbecker, An Easy Programming System, Byte Publications, december 1978.

[3] Lemmens Luc, ST 62 Microcontrollers, Uitgeverij Elektuur, Postbus 75, 6190 AB Beek.

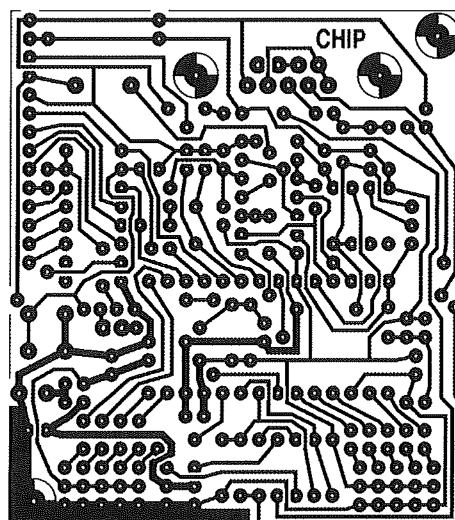
[4] Het I^2C-protocol staat in de data-sheets van de 24C16. Te downloaden van onder andere www.st.com; www.infineon.com; www.atmel.com. Een uitgebreide Nederlandstalige toelichting op het I^2C-protocol is als hoofdstuk 6/10.6 verschenen in "Hobby Elektronica & Actueel IC-handboek", aanvulling 47, te bestellen via www.hobbyelektronica.nu.

[5] Informatie over LCD's is te vinden op www.epemag.wimborne.co.uk, www.senet.com.au/~cpeacock, www.iae.nl (ga naar piazza en kies pouweha).

Bob Stuurman

6.7 Chip, een zelfbouw computertje

6.7 Chip, een zelfbouw computertje



Figuur 4/6.7.1-2: De enkelzijde print voor Chip.

HOE MAAKT U DEZE PRINT?

OPTIE 1: zelf maken

U scant deze pagina en drukt deze met een inkjet-printer af op A4 formaat op transparante folie. U knipt de print uit en belicht er de fotogevoelige printplaat mee.

OPTIE 2: via Internet

Op **www.hobbyelektronica.nu** selecteert u uit het linker menu de optie "Printservice". In het rechter venster selecteert u het hoofdstuknummer. U kunt nu de print als TIF-file downloaden. U opent deze file in een beeldbewerkingsprogramma en drukt deze met de op de Internet-pagina aangegeven afmetingen op transparante folie af. U belicht hiermee de fotogevoelige print.

OPTIE 3: bestellen

U stuurt een **ONGEFRANKEERD** briefje naar Vego VOF, Antwoordnummer 30020, 6374 ED Landgraaf, met vermelding van het hoofdstuknummer. U krijgt per kerende post het printontwerpje op transparante folie **GRATIS** toegestuurd. U belicht hiermee de fotogevoelige print.

6.7 Chip, een zelfbouw computertje

4/6.7.2

De Chip instructieset

Aantal instructies
Een Chip programma bestaat uit een reeks macro-instructies, ieder twee bytes lang. Het programma begint op adres 000h van de externe EEPROM en het hoogste adres van de EEPROM is 7FFh. Een programma kan dus maximaal circa duizend instructies bevatten.

Het commando prog
Met het commando **prog**, zie figuur 4/6.7.2-1, kan het programma in de EEPROM worden gezet, worden bekeken en/of gewijzigd. Het standaard-adres voor **prog** is 000h en oneven adressen worden niet geaccepteerd, een instructie moet altijd op een even adres staan. Opmerking: het hoogste adres dat met **prog** kan worden benaderd is 7FFh, daarboven is het adresgebied echter niet leeg.

De registers
In het gebied van 800h tot en met 8FFh zijn namelijk de registers van de microcontroller geprojecteerd. Die bevinden zich daar natuurlijk niet echt, maar een Chip programma ziet ze daar wél en kan ze lezen en schrijven. Zo is het mogelijk om bijvoorbeeld de auto-reload timer om te programmeren zodat op een van de servo-uitgangen een analoog (PWM) signaal komt te staan.

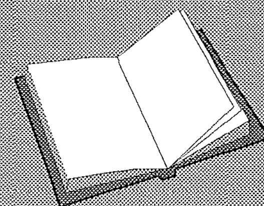
De Chip macrotaal
Iedere instructie bestaat uit twee bytes, maar het is eenvoudiger ze te beschouwen als zijnde opgebouwd uit vier hex cijfers, ook wel nibbles genoemd.

```
MS-DOS Prompt - CT
command?
prog 100

0100 778C +
0102 778D -
0100 778C 1234
0102 778D 5678
0104 8080 -
0102 5678 -
0100 1234 +
0102 5678 +
0104 8080
escaped!
```

Figuur 4/6.7.2-1: Het commando **prog**.

LEES OOK:
Hoofdstuk 4/6.7.1



Een nibble bevat vier bits en kan dus de waardes 0h tot en met Fh bevatten. Een Chip instructie bevat vier nibbles en heeft de algemene vorm van:

AXYB

Hierin geeft A het type instructie aan, X en Y zijn variabelen en B is een precisering van de instructie. Voor X en Y moeten hex cijfers worden ingevuld, dus een cijfer 0h tot en met Fh, dientengevolge kunnen er zestien variabelen worden geadresseerd. Iedere variabele is één byte groot. Behalve instructies voor variabelen zijn er instructies om de loop van het programma te beïnvloeden, voor conversies, voor het display, enz. Als we de instructieset doorlopen, zie de laatste pagina's van dit hoofdstuk, wordt alles duidelijk.

De instructieset nader bekeken
In de volgende paragrafen gaan we de voornaamste Chip instructies aan een nader onderzoek onderwerpen.

Program flow
De instructie **0000** (nop) doet niets en wordt meestal gebruikt om een of meer plaatsen te reserveren of in software vertragingslussen. Met instructie **1MMM** (jump) kan naar ieder adres in het programmageheugen worden gesprongen en met **2MMM** (call sub at MMM) wordt een subroutine op adres MMM aangeroepen.
Een subroutine moet altijd worden beeindigd met **00EE** (return from sub). Er zijn maximaal 8 nesting niveau's mogelijk.
Instructie **0050** (break to operating system) beëindigt de uitvoering van een Chip programma op een gedefinieerde plaats en brengt de gebruiker terug in de commando processor. Ditzelfde is ook mogelijk door drukknop S1 in te drukken, dan echter is de plaats niet bekend.

Set pointer
Vaak is het nodig om variabelen te converteren van en naar decimaal of hexadecimaal. Dit gebeurt dan op de A-stack (arithmetic stack, rekenkundige stack). Dit zijn drie inwendige microcontroller registers. Door middel van een "pointer", die op alle adressen kan worden "gericht" zijn verplaatsingen (feitelijk kopiëren) van en naar variabelen mogelijk evenals naar het LCD en de A-stack en zelfs naar alle registers in de microcontroller.
Omdat de A-stack zo belangrijk is, is er een aparte instructie om de pointer op de A-stack te richten, namelijk **00AA**. MP is de geheugenplaats waar de pointer naar wijst. De instructie **p + 1** zet de pointer 1 positie hoger en de instructie **p + vx** verhoogt de pointer met de waarde van variabele vx. Beide instructies worden als 16 bit optellingen uitgevoerd.
Opmerking: uitsluitend de instructies onder Set pointer kunnen de pointer veranderen.
De instructie **save pointer** slaat een kopie van de pointer intern op en **restore pointer** zet de kopie terug in de pointer.

Pointer conversions on A-stack
Er zijn vier conversie instructies, namelijk:
- variabele naar hexadecimaal;
- variabele naar decimaal;
- hexadecimaal naar variabele;
- decimaal naar variabele.

Voor deze conversies wordt de A-stack gebruikt en daarom moet met de instructie **AA** eerst de pointer daarop wor-

den gezet. Vervolgens kan variabele X naar decimaal worden geconverteerd met **4X13** of naar hexadecimaal met **4X12**. De pointer blijft onveranderd. Stel dat variable A de waarde A7h heeft, dan geeft **4A13** het volgende resultaat:

```
A-stack        31h
A-stack + 1    36h
A-Stack + 2    39h
```

Merk op, dat dit de ASCII-waarden zijn voor respectievelijk 1, 6 en 9, het decimale equivalent voor A7. Omdat het LCD een "ASCII-device" is, kunnen deze cijfers direct op het LCD worden gezet en wel met instructie **DMN3**. Willen we ze op positie 0, 1 en 2 van het LCD, dan wordt de instructie **D023**.
Als we alleen de tientallen en eenheden op het display willen laten zien, kan de pointer met instructie **00A1 (p + 1)** op de tientallen worden gezet en de display instructie **D013** kan worden gebruikt.
Als we nu dezelfde variabele A met A7h naar hex willen converteren, dan gebruiken we instructie **4A12** en het resultaat is:

```
A-stack        41h
A-stack + 1    37h
```

Dit zijn de ASCII-waarden voor Ah en voor 7h. We zetten ze op LCD-positie E en F met instructie **DEF3**.
De instructies **4X31** en **4X21** werken precies omgekeerd, waarbij het wel nodig is om eventuele voorlopende "nullen" op de A-stack te zetten. Dus om bijvoorbeeld het decimale getal 38 te converteren en in variabele C te laden moeten de volgende getallen op de A-stack worden gezet (let op, de maximale waarde van een byte is 255d):

```
A-stack        30
A-stack + 1    33
A-stack + 2    38
```

Na uitvoering van instructie **4C31** bevat variable C de waarde 26h, het hexadecimale equivalent van 38d.

Pointer-Variable moves
Met de instructies **5XY5** en **5XY7** worden één of meer variabelen gekopieerd in het geheugen vanaf MP, dus het adres waar de pointer naar wijst, en vice versa. De pointer zelf wordt hierbij niet veranderd. X is de begin-variabele en Y de eind-variabele (X <= Y).
Als de pointer naar adres 200h wijst, zal instructie **5665** variabele V6 kopiëren op adres 200h. Omgekeerd zal dan instructie **5667** de inhoud van adres 200h kopiëren in V6. Zo kan iedere individuele variabele dus in het geheugen worden gekopieerd en omgekeerd.
Instructie **50F5** zal de variabelen 0-F in het geheugen kopiëren, V0 komt op MP, V1 op MP + 1, enz.
Door instructie **50F7** worden de inhouden van de geheugenplaatsen 200h-20Fh gekopieerd in variabelen 0-F, ervan uitgaande dat MP nog 200h is. Door middel van deze instructies kunnen variabelen in het programmageheugen worden bewaard en teruggezet.
De instructie **5XY7**, tot slot, is uitermate handig om variabelen in één keer vanuit het programmageheugen te initialiseren.

Constants
Hierbij krijgt VX direct de waarde KK (**6XKK**) of wordt KK bij VX opgeteld (**7XKK**). Als de optelling een overflow geeft, dan wordt VF = 01h, anders VF = 00h.

6.7 Chip, een zelfbouw computertje

Als FFh bij de variabele wordt opgeteld, dan wordt deze (door overflow) effectief met 1 verminderd.

Skip instructies

Uiteraard zijn er ook instructies om condities te testen. Hiervoor dienen de skip instructies. Skip betekent “overslaan”, dus als de conditie “waar” is, wordt de erop volgende instructie overgeslagen. Dat zal dan bijna altijd een jump instructie zijn. Het werken met conditionele skips in plaats van met conditionele sprongen is even omdenken, maar na korte tijd is men er volkomen aan gewend. Zo betekent de instructie **9XKK** skip if VX = KK, dus de instructie **9600** betekent: als variabele 6 de waarde 00h heeft, zal de volgende instructie worden overgeslagen.

De instructie **0011** (skip always) is handig om een instructie onder te zetten, die in de normale program flow moet worden overgeslagen, maar wél moet worden uitgevoerd als er via een jump naar toe wordt gesprongen. Als “skip always” onder een conditionele skip wordt gezet, heeft dat tot gevolg dat de conditie wordt omgekeerd, bijvoorbeeld:

```
skip if VX = KK
skip always
jump to MMM
```

heeft tot gevolg dat de sprong wordt uitgevoerd als VX = KK.

Display

Het LCD heeft zestien karakters, het linker is karakter 0h, het rechter karakter Fh. Het display kan direct worden aangestuurd, maar ook indirect, door middel van twee variabelen. De instructie **DMN0** bijvoorbeeld wist het display te beginnen bij karakter M tot en met karakter N. Dus **D330** wist alleen karakter 3 en **D0F0** wist alle karakters van het display. **DXY8** wist het display vanaf het karakter dat in variabele X staat tot dat in variabele Y (X <= Y).

Als de pointer op het begin van een karakterstring wordt gezet, kunnen karakters op het display worden gezet met de instructies **DMN3** of **DXYB.** Het LCD is een “ASCII-device”, met andere woorden karakters die op het display worden gezet dienen conform de ASCII-tabel te zijn.

De instructie **DDDD**, “Rotate text MP on display” laadt de string naar welks eerste karakter de pointer wijst, vanuit het geheugen in het geheugen van het LCD. De string mag maximaal 24 karakters lang zijn en moet eindigen met 00. Vervolgens wordt door het operating system eenmaal per seconde het commando “shift” naar het LCD gestuurd. Voordat de tekst verschijnt is er een geluidssignaal evenals bij elke shift. Dit is voor de gebruiker een volkomen transparant proces. De tekstrotatie wordt beëindigd door instructie **DDDE** die het LCD ook reset.

Timers

Chip bevat vier timers, een alarm timer (tone), een algemene timer en een minuten en seconden timer. Alle timers tellen omlaag tot ze nul zijn. De minuten en seconden timer zijn gekoppeld, dus als de minuten timer één lager wordt, dan wordt de seconden timer geladen met 3Bh (59d). Zolang de alarm timer ongelijk nul is, wordt een geluidssignaal opgewekt. De maximale tijdsduur is 9 seconden. Dit is ook de maximale tijdsduur van de algemene timer. Met instructies **CX00** tot en met **CX03** worden

de timers in variabele X geladen en met instructies **CX10** tot en met **CX13** worden ze vanuit variabele X geladen.

Clock
De real-time clock hoeft alleen te worden gelezen. Hiervoor dienen instructies **CX04** tot en met **CX08**. Door middel van enkele Chip instructies kan een lopende klok op het LCD worden getoond. Ook is het mogelijk om door middel van een tijdstring in het geheugen te controleren of de kloktijd gelijk is aan de stringtijd en hierop actie te ondernemen. Er kunnen meerdere van deze tijdstrings in het geheugen staan.

Random number
Met instructie **0XKK** wordt vanuit de watchdog timer en de predivider van timer1 een random number (toevalsgetal) gegenereerd. Als masker dient KK. Dit wordt met het random number ge-AND waardoor het random number binnen gebieden kan worden begrensd omdat alleen de bits, die in het masker "1" zijn worden doorgelaten en de overige bits "0" worden. Met masker 07h kan het random number de waarden 0h tot en met 7h krijgen, met masker 0Fh de waarden 0h tot en met Fh, maar met masker 08h alleen de waarden 0h of 8h. Let op: Omdat de instructieset een aantal instructies bevat, waarvan de eerste twee cijfers "00h" zijn, mag voor de random number variabele niet V0 worden gebruikt.

Arithmetic
De "A"-groep bevat de logische en rekenkundige bewerkingen. Hierbij zijn altijd twee variabelen betrokken, VX en VY, waarbij VX is te beschouwen als de accumulator, dus de variabele die het resultaat zal bevatten (uitgezonderd **AXY0** en **AXY6**). Bij de instructies **AXY4**, **AXY5** en **AXY7** (en ook **7XKK**) wordt variabele VF gebruikt om de carry, borrow of overflow in te zetten. Na deze instructies zal VF 01h zijn bij een carry, borrow of overflow (anders 00h). De instructies **AXY6** en **AXY7** kunnen worden gebruikt om een waarde te "schalen". Door **AXY6** worden VX en VY met elkaar vermenigvuldigd en als 16 bit getal op de A-stack gezet. Instructie **AXY7** deelt dit 16 bit getal door de 8 bit variabele VY. Als het resultaat van de deling groter is dan FFh en daardoor te groot voor de variabele, wordt de overflow gezet. De instructies **AXY6** en **AXY7** zijn heel eenvoudig te gebruiken. Stel dat V0 een waarde bevat die door een temperatuursensor is gemeten en dat deze waarde met 37h/58h moet worden vermenigvuldigd (geschaald) om een waarde in graden Celsius te krijgen:

```
6137   V1 = 37h
6258   V2 = 58h
00AA   set pointer to A-stack
A016   MP = V0 * V1
A027   V0 = MP/V2
```

Digital I/O
Inwendig bevat Chip 15 uitgangen 0h tot en met Fh, waarvan er 5 (0-4) op connector K1 naar buiten zijn gevoerd. Alle uitgangen kunnen met instructie **E21N** worden "geset" en met instructie **E20N** worden "gereset", waarbij N het nummer van de uitgang is. De stand van iedere uitgang kan door middel van de conditionele skip instructies **E11N** of **E10N** op hoog of laag zijn worden getest. Dit maakt het mogelijk om uitgangen 5h tot en met Eh als bitvlaggen te gebruiken, dus om iets te "onthouden". Bitvlag Fh

6.7 Chip, een zelfbouw computertje

wordt iedere seconde door het operating system gezet. Dat is handig om bijvoorbeeld een real-time klok op het LCD eenmaal per seconde te refreshen (waarbij bitvlag Fh natuurlijk wel moet worden gereset).
De vijf (0-4) digitale ingangen kunnen eveneens door middel van conditionele skips worden uitgelezen.

Analog I/O
De F-groep is voor de analoge I/O. Dit betreft dus het inlezen van een spanning van de ingangen en het sturen van de servo's. Instructie **F3F2** leest van ingang 2 de analoge waarde en zet deze in V3. De maximale waarde is FFh en de minimale waarde is 00h, respectievelijk de spanningen op de V_{dd} en Gnd aansluitingen van de microcontroller.
Instructie **FX0N** doet hetzelfde, maar voegt er een extra bewerking aan toe, namelijk de hoge nibble van VX wordt verplaatst naar de lage nibble en de hoge nibble krijgt de waarde 3. Zo krijgt variabele X de ASCII-waarde die overeen komt met een ingedrukte toets van het Chip keyboard. Als er geen toets is ingedrukt krijgt VX de waarde 3Fh.
Standaard is de servo drive uitgeschakeld, met **FEE1** wordt de pulsopwekking aangezet en met **FEE0** uitgezet. Met **FXE2** en **FXE3** wordt de pulstijd ingesteld voor respectievelijk servo 1 en servo 2.

Chips foutafhandeling
Chip bevat een foutafhandelingssysteem, dat foutieve instructies opvangt. Als een fout wordt ontdekt, wordt op het scherm een foutmelding gegeven met opgave van het adres waar de fout is gevonden. De uitvoering van het Chip programma wordt beëindigd en er wordt teruggekeerd naar de commando processor.

Uitzonderingen
Omdat voor rekenkundige instructies variabele VF als carry, borrow of overflow vlag wordt gebruikt, is het is raadzaam om in een programma de variabele VF niet als normale variabele te gebruiken. Datzelfde geldt voor output F, die eenmaal per seconde door het operating systeem wordt gezet en die kan worden gebruikt om een bepaalde actie eenmaal per seconde uit te voeren.
Voor het genereren van een random number mag V0 niet worden gebruikt. Dat zou tot onvoorspelbare fouten leiden.

Bob Stuurman

6.7 Chip, een zelfbouw computertje

Chip instruction set

Program flow		Mnemonic
0000	nop	nop
1MMM	jump to MMM	jp address [label]
2MMM	call sub at MMM	call address [label]
00EE	return from sub	ret
0050	break to operating system	break

Set pointer		Mnemonic
00AA	set pointer to A-stack	p = a-stack
3MMM	set pointer to MMM	p = address [label]
00A1	pointer + 1	p + 1
4X00	pointer + VX	p + vx
0020	save pointer	save p
0021	restore pointer	rest p

Pointer conversions on A-stack		Mnemonic
4X13	VX -> 3 decimals MP	vx to 3dec mp
4X31	VX = 3 decimals MP	vx = 3dec mp
4X12	VX -> 2 hex MP	vx to 2hex mp
4X21	VX = 2 hex MP	vx = 2hex mp

(Pointer not changed, MP = MSD)

Pointer-Variable moves		Mnemonic
5XY5	VX...VY -> MP	vx,vy to mp
5XY7	MP -> VX...VY	vx,vy = mp

Constants		Mnemonic
6XKK	VX = KK vx = kk	
7XKK	VX = VX + KK (VF = carry)	vx + kk

Skips		Mnemonic
0011	skip always	skip a
8XKK	Skip if VX <> KK	skip vx <> kk
9XKK	Skip if VX = KK	skip vx = kk
BXY0	Skip if VX = VY	skip vx = vy
BXYF	Skip if VX <> VY	skip vx <> vy
BXY7	Skip if VX < VY	skip vx < vy
BXY8	Skip if VX > VY	skip vx > vy

Display		Mnemonic
DMN0	Clear display from M to N	cd m,n
DXY8	Clear display from VX to VY	cd vx,vy

6.7 Chip, een zelfbouw computertje

```
DMN3   Load display from M to N       ld m,n
       (from MP)
DXYB   Load display from VX to VY     ld vx,vy
       (from MP)
DDDD   Rotate text MP on display      rotate
       (string must end with 00, max 24d characters)
DDDE   stop rotate, clear display     stop rotate
```

Timers — **Mnemonic**

```
CX00   VX = Alarm timer               vx = tone
CX01   VX = Timer                     vx = timer
CX02   VX = Seconds timer             vx = sectimer
CX03   VX = Minutes timer             vx = mintimer
CX10   VX -> Alarm timer              vx to tone (1C=1s)
CX11   VX -> Timer                    vx to timer (1C=1s)
CX12   VX -> Seconds timer            vx to sectimer
CX13   VX -> Minutes timer            vx to mintimer
```

Clock — **Mnemonic**

```
CX04   VX = Weeks                     vx = weeks
CX05   VX = Days                      vx = days
CX06   VX = Hours                     vx = hours
CX07   VX = Minutes                   vx = minutes
CX08   VX = Seconds                   vx = seconds
```

Random number — **Mnemonic**

```
0XKK   VX = Random number             vx = rnd,kk
       (KK is .AND. mask)
Note:  do not use V0
```

Arithmetic — **Mnemonic**

```
AXY0   VX -> VY                       vx to vy
AXY1   VX = VX .AND. VY               vx and vy
AXY2   VX = VX .OR. VY                vx or vy
AXY3   VX = VX .XOR. VY               vx xor vy
AXY4   VX = VX + VY (VF = carry)      vx + vy
AXY5   VX = VX - VY (VF = borrow)     vx - vy
AXY6   MP = VX * VY                   vx * vy to mp
AXY7   VX = MP/VY (VF = overflow)     vx = mp / vy
Note:  use the a-stack for AXY6 and AXY7
```

Digital I/O — **Mnemonic**

```
E20N   Reset output N                 res out n
E21N   Set output N                   set out n
E10N   Skip if output N = 0           skip out n = 0
```

6.7 Chip, een zelfbouw computertje

```
E11N   Skip if output N = 1            skip out n = 1
E00N   Skip if input N = 0             skip in n = 0
E01N   Skip if input N = 1             skip in n = 1
Notes  there are 5 inputs (0-4) and 16 outputs from wich 5
       (0-4) have pins, 5-E can be used as bitflags.
       Outputs are default 0.
       Output F is set every second by the operating system.
```

Analog I/O — **Mnemonic**

```
FXFN   VX = analog input N             vx = ana n
FX0N   VX = keyboard input N           vx = key n
FEE0   Stop servo drive                soff
FEE1   Start servo drive               son
FXE2   VX -> servo 1                   vx to s1
FXE3   VX -> servo 2                   vx to s2
Notes: digital inputs are switched to analog during
       conversion and the pull-up resistor (appr. 200 kOhm)
       is disabled. Keyboard values are 30h-3bh, 3fh = no
       key pressed. Default servo values are 80h
```

Chip assembler directives

```
; This is a comment at the beginning of a line
      org address                      ; comment midline
label equ address
      bytes 001122....ff
      asciz "Hi, I'm Chip!"
Notes: an odd number of bytes will be concluded with 00
       asci-zero will be concluded with 00 or 0000
```

Positioning of characters on display

```
character 0 ............. character f
```

Memory model

```
      08ffh
        |     st62 internal
      0800h
      07ffh
        |     chip programs
      0000h
```

6.7 Chip, een zelfbouw computertje

4/6.7.3

Assembler, voorbeelden en keyboard

De assembler

Inleiding

De Chip assembler is gemaakt om het schrijven van Chip programma's zo eenvoudig mogelijk te maken. De assembler genereert de macro instructies en berekent de sprongen en pointer posities. De assembler doet dit aan de hand van de mnemonics, die in de Chip instructieset staan.
Een mnemonic is een verkorte schrijfwijze van een instructie, die gemakkelijk is te onthouden en die de instructie eenduidig aangeeft. Bij veel instructies wordt een variabele geladen of wordt een kopie van de variabele in een geheugenadres geschreven. Om dit onderscheid eenduidig vast te leggen en ook omdat het gemakkelijk is te onthouden, wordt voor het laden van een variabele de schrijfwijze:

```
VX = ...
```

gebruikt en voor het laden van een adres vanuit een variabele:

```
VX to ...
```

Het Engelse woord "to" betekent "naar", het geeft eenduidig een bestemming aan.

DOS tekstverwerker

Voor het schrijven van Chip programma's in assembly is een tekstverwerker nodig die gewone DOS-tekst maakt, dus er geen opmaakcodes of iets dergelijks tussenvoegt. Programma's als Word of WordPerfect zijn absoluut verboden! Windows Kladblok is bruikbaar, maar een kleine DOS teksteditor, zoals bijvoorbeeld EDT (Norton Editor), is veel handiger. Ook het Windows shareware programma TextPad is zeer geschikt.

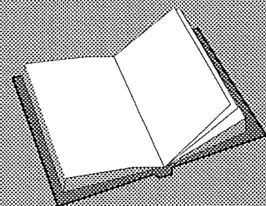
Samenstelling assembler code

Een regel in de Chip assemblertaal is als volgt opgebouwd:

```
label    mnemonic    opfield
```

Het label dient voor de adresaanduiding, het eerste karakter moet a - z of A -

LEES OOK:
Hoofdstuk 4/6.7.1
Hoofdstuk 4/6.7.2
www.vego.nl/chip



Z zijn. De lengte van een label is maximaal acht karakters. Als een label niet nodig is, moet het eerste karakter van de regel een spatie of een tab zijn. De mnemonics kunnen worden gehaald uit de Chip instructieset.
De spaties in de mnemonics mogen eventueel worden weggelaten. Achter de mnemonic kan, na een spatie of een tab, een opfield staan. Bij Chip zal dat altijd een adres of een adres- label zijn.

Assembler aanwijzingen
Er zijn vijf assembler aanwijzingen (directives), die een speciale betekenis voor de assembler hebben.

";"
Als het eerste karakter van een regel ";" is, dan wordt die regel als een commentaarregel beschouwd. Ook kan na de mnemonic of, indien aanwezig, het opfield, na ";" commentaar worden opgenomen.

org adres
De tweede aanwijzing is org adres. Dit heeft tot gevolg dat het programma(deel), dat op "org" volgt, begint op "adres". De regel met "org" wordt na assemblage een commentaarregel en mag daarom geen label bevatten.

label equ adres
De derde aanwijzing is label equ adres. Hier wordt aan een label een adres toegewezen.
Dat is bijzonder handig als met interne registers van de microcontroller wordt gewerkt. In plaats van het lastig te onthouden adres kan de naam van het interne register worden gebruikt. Deze regel wordt ook een commentaarregel in het hex-bestand.

bytes 00112233.....
De vierde aanwijzing is bytes 00112233...... Dit heeft tot gevolg dat de bytes 00h, 11h, 22h, 33h, enzovoort in het programma worden opgenomen. Als de laatste byte op een even adres staat wordt het erop volgende oneven adres opgevuld met 00h.

asciz "Hi I'am Chip!"
De vijfde en laatste aanwijzing is asciz "Hi I'am Chip!". Hier wordt de tekst die tussen de dubbele aanhalingstekens staat omgezet in het ASCII-equivalent in bytes. De assembler sluit de string af met 00 of 0000, om de volgende instructie op een even adres te kunnen zetten. Het sluitkarakter 00 is nodig voor tekst die op het display moet roteren.

Opmerking over bestanden
Het achtervoegsel van een in Chip assembly geschreven programma moet altijd .asm zijn, anders wordt het niet door de assembler herkend.

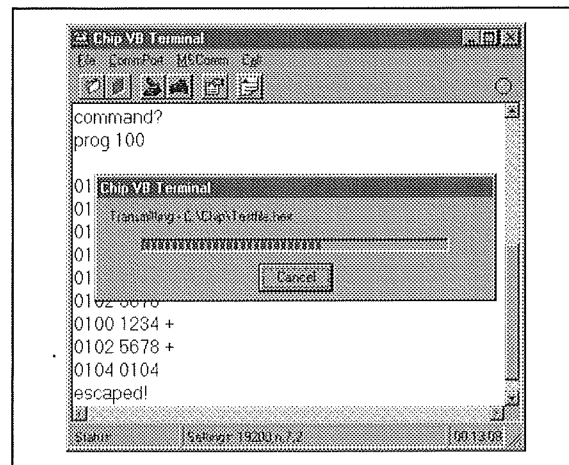
Chipasm.exe
De assembler CHIPASM.EXE wordt gestart met Chipasm naam. Hierin is "naam" de naam van het te assembleren programma. Het achtervoegsel .asm mag worden weggelaten.
Als door de assembler geen fouten worden geconstateerd, dan wordt een .hex bestand gegenereerd, waarvan het voorvoegsel hetzelfde is als van het .asm bestand. Als er wel fouten worden geconstateerd, dan wordt een melding van de aard van de fout en het regelnummer op het scherm gezet.
Het .hex bestand is vrijwel identiek aan het .asm bestand, met dien verstande dat nu de adressen en macro's zijn ingevuld en de regels met org en equ commen-

taarregels zijn geworden. Als het programma wordt afgedrukt, kan het met commando "prog" in de EEPROM worden gezet of worden ge-upload met het terminalprogramma Chipterm.exe.

Opmerking
Het is handig als Chipasm.exe en Chipterm.exe in één map worden gezet, bijvoorbeeld CHIP, waarin ook de Chip programma's komen te staan. In de map kan ook de DOS-tekstverwerker worden gezet, dat is wel zo makkelijk. Wij hebben Chipasm.exe en Chipterm.exe hernoemd in CA.exe en CT.exe. Bovendien is de DOS-tekstverwerker (EDT.exe) in een batchbestand (ED.bat) opgenomen, dat het achtervoegsel .asm achter de naam voegt.

Chip VB Terminal
In plaats van het compacte (ca. 50 kB) en snelle DOS-programma "Chipterm" kan ook het Windows-programma "Chip VB Terminal" worden gebruikt (Windows versies vanaf 95). Dit programma is het bij Visual Basic 6 geleverde terminal voorbeeldprogramma, waarvan de tekstverzending is aangepast voor de verzending van Chip HEX-bestanden, zie figuur 4/6.7.3-1. De instelling van de vertraging geschiedt met een schuifregelaar in het Properties tabblad.
Het programma ziet er wel mooier uit dan ChipTerm, maar voor een functionaliteit die nagenoeg gelijk is, doet het een veel grotere aanslag op de computerbronnen. Na downloaden kan het worden geïnstalleerd door het Setup-programma.

Opmerking
Zowel ChipTerm als Chip VB Terminal hebben de mogelijkheid om .log files te gebruiken. Hierin kan de inhoud van Chip's EEPROM worden gekopieerd (door in "prog" de "+"-toets ingedrukt te houden). Zo kan Chip ook als datalogger worden gebruikt, waarbij de gegevens op de computer kunnen worden verwerkt.



Figuur 4/6.7.3-1: Chip VB Terminal tijdens de verzending.

Testfile.asm
Het bestand Testfile.asm (niet hier afgedrukt, alleen te downloaden van de speciale Internetpagina, zie later in dit hoofdstuk) is geen Chip-programma; de inhoud van ieder adres is gelijk aan het adres zodat, voor alle zekerheid, eenvoudig kan worden gecontroleerd of het uploadproces foutloos geschiedt. Bij lange bestanden als Testfile.asm is het mogelijk, dat niet elke upload foutloos is. Probeer het opnieuw als er een checksumfout is.

Uploaden
Met de 8 bit SPI is een transmissieformaat van maximaal zeven bits mogelijk, omdat bit 7 nodig is als startbit. Alle gegevens worden daarom als hexadecimale karakters naar Chip verzonden.

6.7 Chip, een zelfbouw computertje

```
; Listing First.asm
; Programma om de call en break instructies te demonstreren,
; het gebruik van de assembler aanwijzingen org en asciz en de
; display instructie ld m,n
;
;
start       call showtxt        ; call sub showtxt
            break               ; break to op system
;
;
; subroutine showtxt
;
            org 50              ; start address of sub
showtxt     p = showtxl         ; point p to text string
            ld 0,f              ; load display from char 0 to f
            ret                 ; return from sub
showtxl     asciz "My first program"
;
; end of First.asm
```

Figuur 4/6.7.3-2: De listing van First.asm

Eerst de vier cijfers van het adres, dan de vier cijfers met de inhoud van dat adres, precies zoals ze in de HEX-listing staan. Steeds als er een groep van vier karakters is ontvangen worden deze geconverteerd naar twee bytes en intern opgeslagen. Dan berekent Chip de controlesom en stuurt die naar het terminalprogramma. Na ontvangst van een karakter heeft Chip enige tijd nodig om dat in een buffer te zetten en de SPI opnieuw in te stellen. Het terminalprogramma moet tussen het verzenden van de karakters daarom een korte pauze opnemen in de grootte-orde van enkele milliseconden. Deze vertraging wordt verkregen door een "for next" lus en is afhankelijk van de snelheid van de computer. Chipterm.ini bevat het vertragingsgetal, dat moet worden aangepast (zo klein mogelijk moet worden gemaakt) voor een foutloze transmissie. Het uploaden gaat best wel snel, het schrijven van alle (2048) EEPROM-cellen duurt ongeveer dertig seconden.

Voorbeelden

First.asm
Het programma in de listing van figuur 4/6.7.3-2 (First.asm) geeft een voorbeeld van de opbouw van een Chip programma. Wij gebruiken in Chip programma's uitsluitend onderkast (kleine letters) behalve soms in commentaarregels.
Als het programma is geladen, kan het met CA worden geassembleerd.

CT.exe
Om het hex-bestand in Chip te laden moet CT worden gestart. Als CT voor het eerst wordt gestart, wordt om de te gebruiken COM-poort gevraagd. Deze wordt opgeslagen in het bestand Chipterm.ini.

Belangrijk
In dit bestand staat ook een getal, voor de pauze tussen het versturen van de hex-getallen tijdens het uploaden.

6.7 Chip, een zelfbouw computertje

```
; Listing Charset.asm
; display characterset of the LCD
;
;
            p = charse2             ; point p to text
            ld 0,f                  ; load display
            p = a-stack             ; point p to a-stack
            v0 = 00                 ; v0 is counter
charsel     skip out f = 1          ; wait till out f becomes 1
            jp charsel              ; not yet 1, jump back
            res out f               ; out f = 1, reset out f
            v0 to 2hex mp           ; convert v0 t0 2 hex
            ld 5,6                  ; load display with hex char's
            v0,v0 to mp             ; copy v0 into mp
            ld f,f                  ; and load mp into display pos f
            v0 + 01                 ; update counter
            jp charsel              ; jump back into wait loop
charse2     asciz "Byte    is LCD   "
```

Figuur 4/6.7.3-3: De listing van Charset.asm.

De hardware timer van de PC is daarvoor niet fijnmazig genoeg. Pas dit getal aan tot het uploaden zo snel mogelijk gaat, zonder foutmeldingen. Gebruik daarvoor de DOS-tekstverwerker. Als tijdens het uploaden een fout optreedt, stopt het uploaden en hangt Chip. Druk op S1 en probeer het opnieuw.
Door in CT de menukeuze ALT+l te kiezen, verschijnt een venster waarin de naam van het *.HEX bestand kan worden opgegeven. Het achtervoegsel mag worden weggelaten. Na het drukken op Enter wordt het bestand naar Chip gestuurd. Met het commando "Chip" wordt het gestart.

Waar is de poort gebleven?
Soms ziet een, in een Windows DOS-venster, draaiend programma de seriële poort niet.
Neem dan in system.ini onder [386Enh] de aanwijzing:

ComNAutoAssigned=0

op, waarbij N de gebruikte COM-poort is.

Rechtstreeks adressen wijzigen
We zouden natuurlijk in First.asm de mnemonic "ld 0,f" kunnen veranderen in "rotate" om de tekst te laten roteren, opnieuw assembleren en uploaden, maar het kan ook anders.
Als we het bestand First.hex in de tekstverwerker openen, zien we immers op adres 0052h de instructie D0F3 staan, waarmee de tekst op het display wordt gezet. Met "prog 52" komen we op dit adres en kunnen direct "DDDD" invoeren en het programma starten.
En inderdaad, de tekst roteert nu op het display!

Charset.asm
We willen ook graag weten hoe de ASCII-karakterset van het display er precies uitziet. Vooral de extended karakters interesseren ons, omdat die per LCD kunnen verschillen.

6.7 Chip, een zelfbouw computertje

```
; Listing Hexval.asm
; show analog voltage on input 0 once a second on the LCD display.
; use output flag f, which is set every second by the operating system.
;
;
hexval      p = hexval2             ; point p to display text
            ld 0,f                  ; load initial display
            p = a-stack             ; point p to a-stack
hexval1     skip out f = 0          ; wait for f to be set
            call hexsub             ; f is set, call hexval sub
            jp hexval1              ; and loop
hexval2     asciz "HEX value =        "
;
hexsub      res out f               ; reset seconds flag
            v1 = 04
            v1 to tone
            v0 = ana 0              ; get voltage from input 0 into v0
            v0 to 2 hex mp          ; convert into 2 hex chars
            ld c,d                  ; put 2 hex chars on display
            ret                     ; and return
```

Figuur 4/6.7.3-4: De listing van Hexval.asm.

We willen bijvoorbeeld weten welk teken we het best voor "°" kunnen gebruiken voor de temperatuurmeting. In de listing van figuur 4/6.7.3-3 is het programma Charset.asm te zien dat precies dát doet. Eerst zetten we de pointer op de asciz tekst en laden het hele display. Dan wordt de pointer op de A-stack gezet en v0 wordt 00 gemaakt. We gebruiken output f om een keer per seconde een aflezing te krijgen, want output f wordt iedere seconde door het Operating System gezet. In een lus wachten we dus rustig tot output f wordt gezet en zodra dat het geval is, wordt de jump geskipt (overgeslagen) en kunnen we aan het werk. Eerst moet natuurlijk output f worden gereset en dan wordt v0 omgezet in twee hex-cijfers, die met "ld 5,6" op positie 5 en 6 op het display worden gezet. Nu kopiëren we v0 naar mp en laden displaypositie f vanuit mp. Nadat v0 met 1 is verhoogd, is het werk gedaan en springen terug in de wachtlus. We assembleren het programma, laden het in Chip en starten het. Bekende en vreemde karakters passeren ons oog. Die zullen wel Japans zijn, want de LCD-controller komt uit Japan. Na even wachten zien we gelukkig een karakter verschijnen dat we prima als gradenteken kunnen gebruiken. Ons werk heeft zich geloond en we drukken op S1 om naar de commandoprocessor terug te keren!

Labels benoemen?

Weliswaar werkt "charset" perfect, maar toch komt misschien de vraag op, waarom de labels geen namen hebben als "wait" en "tekst"? Wel, het is beter om als labelnaam de naam van de (sub)routine te gebruiken in combinatie met een volgnummer (0...9, A...Z). Vooral in grote programma's, met veel labels, wordt de structuur veel duidelijker en is de kans op fouten door dubbele labels kleiner.

Hexval.asm

Voor veel toepassingen moet de gebruiker gegevens kunnen invoeren.

6.7 Chip, een zelfbouw computertje

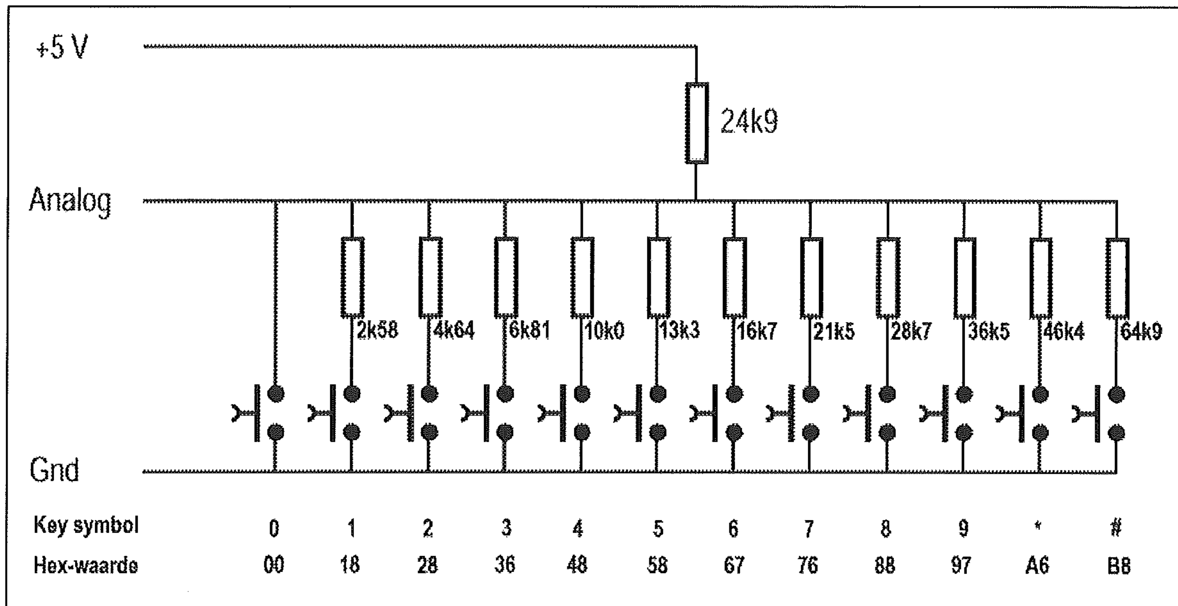
```
; Listing Servopot.asm
; rotate servo 1 by turning pot connected to input 1
;
;
            son                 ; start servo drive
            p = servop2         ; point p to text
            ld 0,f              ; load text to lcd
            p = a-stack         ; point p to a-stack
servop1     v0 = ana 1          ; v0 is voltage from input 1
            v0 to s1            ; load v0 into servo 1
            skip out f = 1      ; skip if seconds flag out f
            jp servop1          ; jump start ofloop
            res out f           ; reset seconds flag
            v0 to 3dec mp       ; convert v0 into 3 decimals mp
            ld 9,b              ; load lcd positions 9,b
            jp servop1          ; jump to start of loop
servop2     asciz "potval = ??? dec"
```

Figuur 4/6.7.3-5: De listing van Servopot.asm.

We denken bijvoorbeeld aan een temperatuur, het aantal NiCad-cellen of het aantal minuten en seconden voor een goed gekookt eitje. We hebben een keyboard nodig. Er is maar één mogelijkheid om het keyboard op Chip aan te sluiten en dat is op een input. Vooraf moeten we testen of dat wel mogelijk is, want anders doen we werk voor niets. Aan een snoertje met drie gekleurde aders sluiten we een 3-polige female SIL-header aan, voor de verbinding met input 1. We werken de verbindingen netjes af met krimpkousjes want het snoertje zal ons zeker vaker van pas komen. Op de open kant sluiten we een potentiometer van 10 kΩ aan, zodanig dat de spanning op de in-pen regelbaar is. De weerstandswaarde van de potentiometer is niet echt belangrijk, alles tussen 1 kΩ en 100 kΩ is bruikbaar.
Nu komt het programma "Hexval.asm" (zie de listing van figuur 4/6.7.3-4) prima van pas. Ook hier maken we gebruik van de secondenvlag "output f", voor een duidelijke weergave op het betrekkelijk langzame display. Voor de eigenlijke meting zorgt een subroutine onder de skip, zodat nu de opdracht "skip out n = 0" moet worden gebruikt.
We assembleren het programma, laden en starten het. Op het display verschijnt een waarde. We verdraaien langzaam de potentiometer en wat blijkt, het is mogelijk alle waarden van 00h tot en met FFh te meten. De proef is prima geslaagd. Het keyboardje moet lukken.

Servopot.asm
Nu we toch een potentiometer hebben aangesloten willen we onderzoeken of daarmee een servo kan worden gestuurd. Op servo 1 sluiten we een Futaba S3003 servo aan, een goedkope en goede servo.
In de listing van figuur 4/6.7.3-5 staat het programma Servopot.asm. Met "son" starten we de servopuls, dan laden we het display met de tekst en zetten de pointer op de A-stack.

6.7 Chip, een zelfbouw computertje



Figuur 4/6.7.3-6: Het schema van het toetsenbord.

Dan meten we de potentiometerspanning in v0 en kopiëren v0 naar s1. Als de secondenvlag niet is gezet blijven we in de lus. Als de secondenvlag wel staat, wordt deze gereset, waarna v0 wordt geconverteerd naar 3 decimale getallen. Deze worden op het display gezet en met een jump springen we terug in de lus.
Het programma werkt goed. Met de potentiometer kan de servo heel precies worden geregeld. Deze kennis zal zeker van pas komen bij de bouw van de robot.

Het keyboard

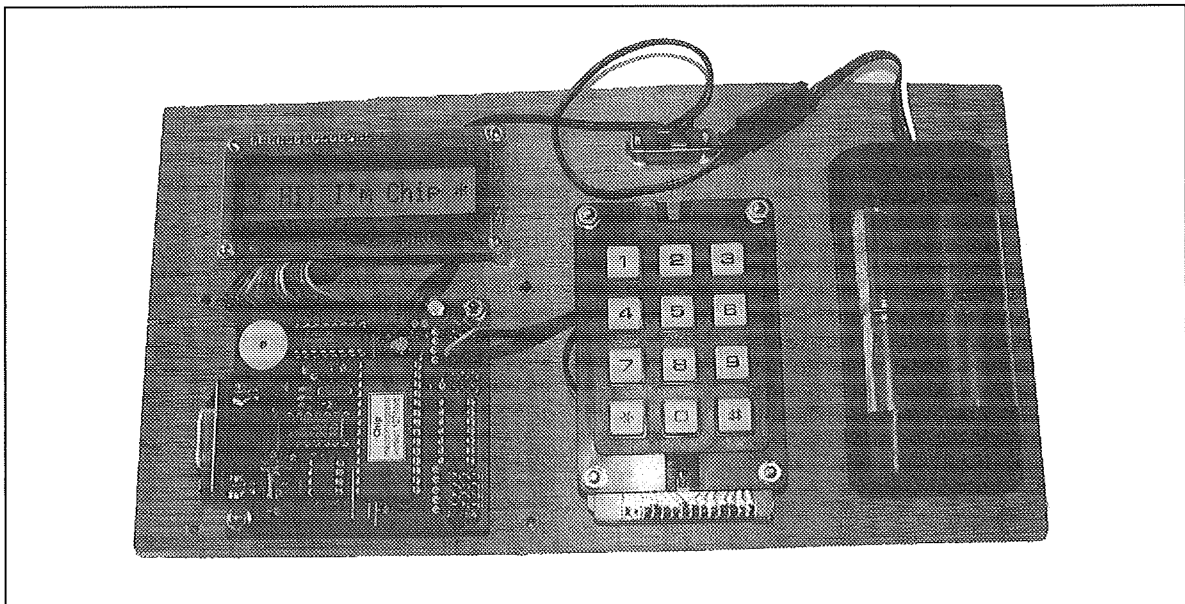
Inleiding
Als keyboard gebruiken we een type met 12 toetsen en wel 0-9, * en #. De toetsen hebben een gemeenschappelijke aansluiting en iedere toets heeft een eigen aansluiting. Zo is het mogelijk om iedere toets in een spanningsdeler op te nemen en bij indrukken een eigen, unieke spanning te laten afgeven. De schakeling van het keyboard is getekend in figuur 4/6.7.3-6. In de tabel eronder staan de namen van de toetsen en de hex-waarde van de spanning die bij indrukken wordt gemeten.

Montage
De weerstanden zijn aan de onderkant van het keyboard gemonteerd met aan de achterkant een drie-aderig snoertje met daaraan een connector, precies zoals het snoertje voor de potmeter.
In het schema staan 1 % weerstandswaarden, maar persé nodig is dat niet. Het belangrijkste is dat de lage nibble van iedere toets ligt tussen 5h-Ah, dan is de tolerantie voldoende.
Met het programma Hexval.asm kan voor iedere toets de weerstandswaarde proefondervindelijk worden bepaald. Bij gebruik van 5 % weerstanden zal het soms nodig zijn om twee weerstanden in serie te schakelen, maar het lukt op deze manier altijd om een passende hexwaarde te krijgen.

6.7 Chip, een zelfbouw computertje

```
; Listing Keytest.asm
; test command vx = key n
;
;
            va = 01                 ; va is key beep
            p = keytes3             ; load display
            ld 0,f
            p = a-stack             ; point p to a-stack
keytes1     v0 = key 0              ; get key value
            v0 to 2hex mp           ; convert to hex...
            ld c,d                  ; ...and put on display
            skip v0 <> 3f           ; skip on key press
            jp keytes1              ; loop
            va to tone              ; give beep
keytes2     v0 = key 0              ; get key value
            skip v0 = 3f            ; skip on no key press
            jp keytes2              ; loop untill key release
            jp keytes1              ; jump for next key press
keytes3     asciz "key value =     h"
```

Figuur 4/6.7.3-7: De listing van Keytest.asm.



Figuur 4/6.7.3-8: Het Chip experimenteersysteem.

Keytest.asm

Zoals we eerder hebben gezien is Chip "ASCII-georiënteerd", dus moeten ook de toetsen een ASCII-waarde afgeven: 30h tot en met 39h, 3Ah en 3Bh. Dat is nu precies wat de instructie "vx = key n" doet. Als er een toets is ingedrukt, wordt die ontdenderd en nogmaals gemeten. Met het programma Keytest.asm in figuur 4/6.7.3-7 kan worden getest of het keyboard goed werkt. Als er een toets wordt ingedrukt wordt er een piepje ge-

geven en de ASCII-waarde in hex op het display gezet. Dan wordt gewacht tot de toets wordt losgelaten.

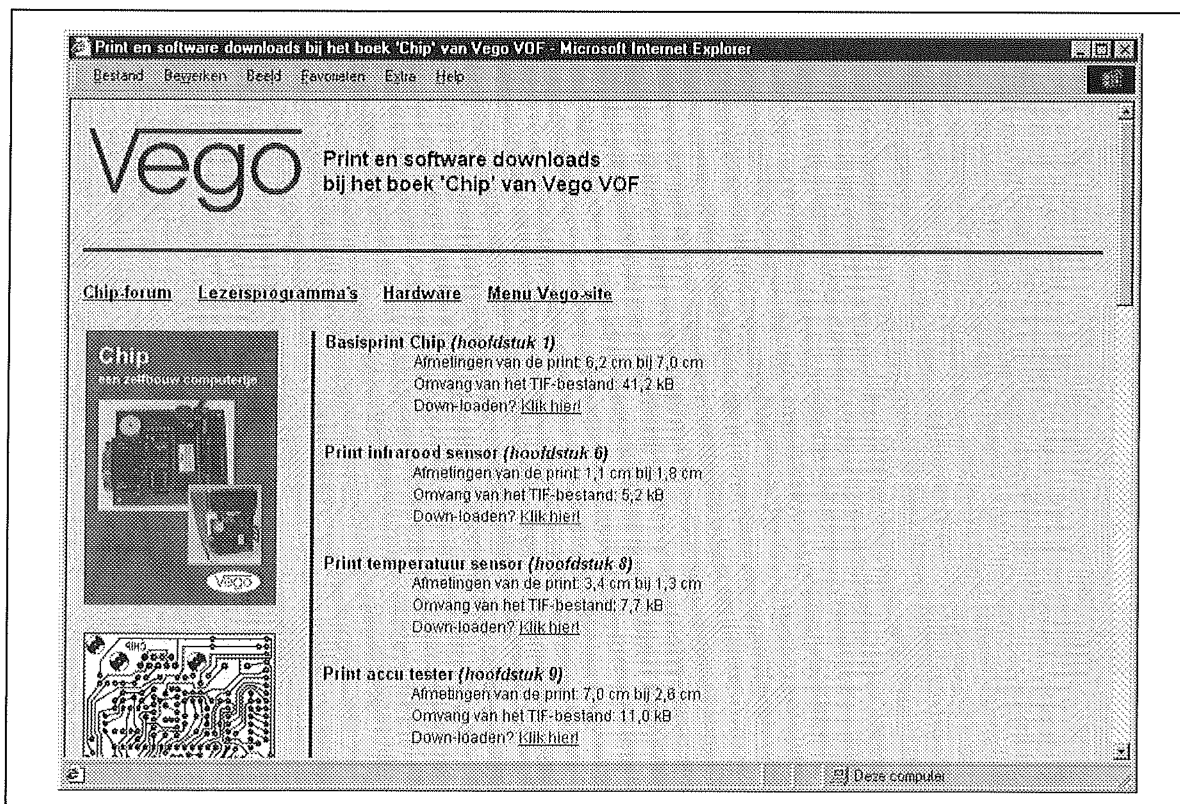
Opmerking
Dit voorbeeld werkt goed, maar er zijn situaties waarbij niet gewacht mag worden omdat de hoofdlus van het programma altijd door moet draaien. Ook dat is op te lossen zoals we in een later hoofdstuk zullen zien.

Experimenteerbord
Ondertussen is het misschien een goed idee om een experimenteersysteem te maken door Chip, het LCD en het keyboard op een plankje te monteren, zie figuur 4/6.7.3-8. Op het plankje kan ook de NiCad-accu een plaatsje vinden.

Van dit systeem zullen we veel profijt hebben bij onze verdere ontwikkelingen.

Chip op Internet

Inleiding
Een naslagwerk als “Hobby Elektronica & Actueel IC-handboek” is nog steeds een onvervangbare datadrager, maar een boek heeft natuurlijk ook nadelen. Zeker als een boek gaat over moderne elektronica, zoals dít boek, loopt de lezer het risico dat de informatie voor een deel alweer verouderd is.
Vandaar dat de uitgever het unieke initiatief heeft genomen om een Internetpagina aan het project “Chip” te koppelen.



Figuur 4/6.7.3-9: De hoofdpagina van Chip.

6.7 Chip, een zelfbouw computertje

Via deze Internetpagina wordt u, lezer van dit naslagwerk, actueel op de hoogte gehouden van nieuwe ontwikkelingen, kunt u de beschreven software downloaden en kunt u de layout's van de printen ophalen.

www.vego.nl/chip
Als u in uw favoriete Internetbrowser www.vego.nl/chip intoetst, komt u terecht op de openingspagina van Chip, "Print en software downlaods bij het boek Chip van Vego VOF", zie figuur 4/6.7.3-9. Van deze pagina kunt u de printen en de software downloaden en kunt u naar drie subpagina's:
- Chip-forum;
- Lezersprogramma's;
- Hardware.

Downloaden van de printontwerpen
Via de genoemde pagina kunt u dus de printen downloaden. Dat gaat heel eenvoudig. Ga met de muiscursor op de link "Klik hier!" staan en klik op de rechter muisknop. In het pop-up venster dat nu verschijnt, zie figuur 4/6.7.3-10, klikt u met de linker muisknop op de optie "Doel opslaan als...". U kunt vervolgens het ontwerp van een van de printen opslaan in een door u geselecteerde directory (map) van uw harde schijf.

Figuur 4/6.7.3-10: Het downloaden van de printontwerpen van "Chip".

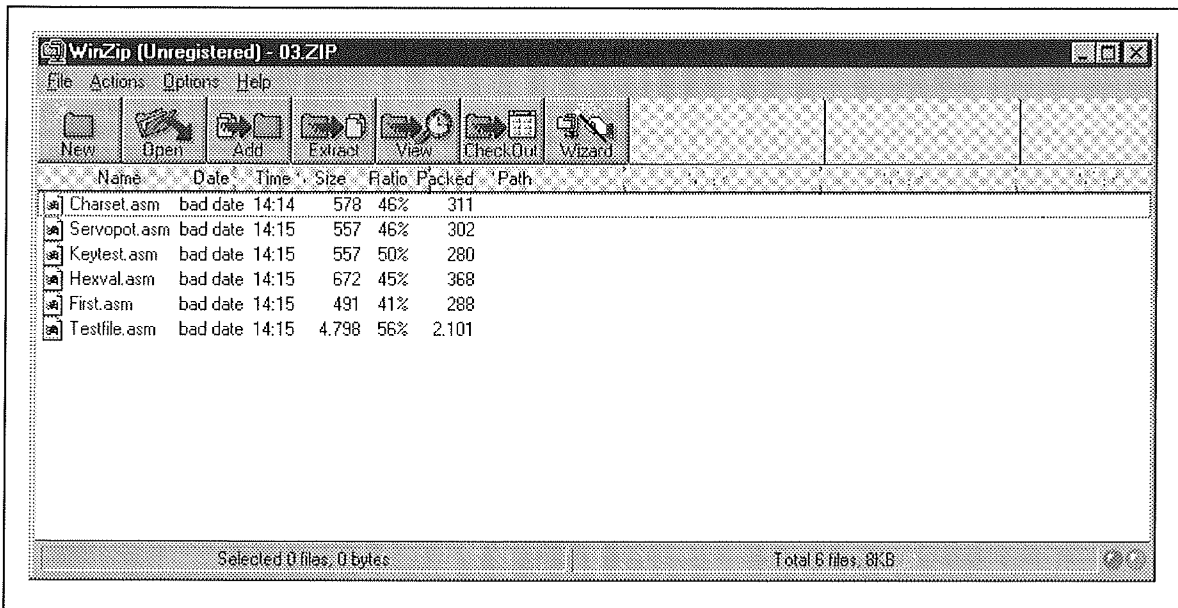
Afdrukken van de printontwerpen
Het gedownloade printbestand heeft het TIF-formaat, een formaat dat u in ieder grafisch programma kunt openen en afdrukken. Wij werken met PaintShop Pro, zie figuur 4/6.7.3-11, waarin we als voorbeeld de basisprint van "Chip" hebben geopend.

Figuur 4/6.7.3-11: De basisprint van "Chip" in PaintShop Pro.

Figuur 4/6.7.3-12: Het instellen van de afmetingen van de print.

Via het venster "Page Setup", zie figuur 4/6.7.3-12, kunt u de afmetingen van de print instellen op de goede afmetingen.

6.7 Chip, een zelfbouw computertje

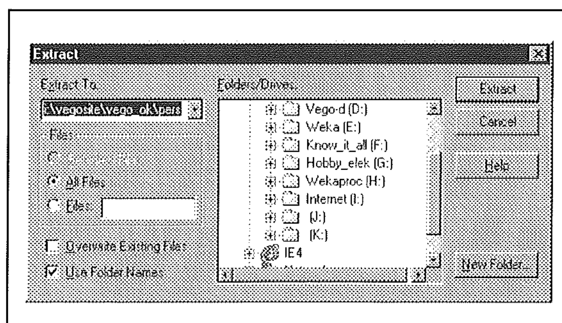


Figuur 4/6.7.3-13: De programmabestanden van dit hoofdstuk zitten in het bestand 03.ZIP.

Deze zijn zowel in dit naslagwerk als op de Internetpagina vermeld.
In het geval van de basisprint zijn die afmetingen 62 mm breed en 70 mm hoog.
U print vervolgens de print af op uw inkjet printer en wel op de speciale transparante folie die tegenwoordig in iedere kantoorboekhandel te koop is. Met deze folie kunt u de print op de gebruikelijke manier belichten op ultraviolet gevoelige printplaat.

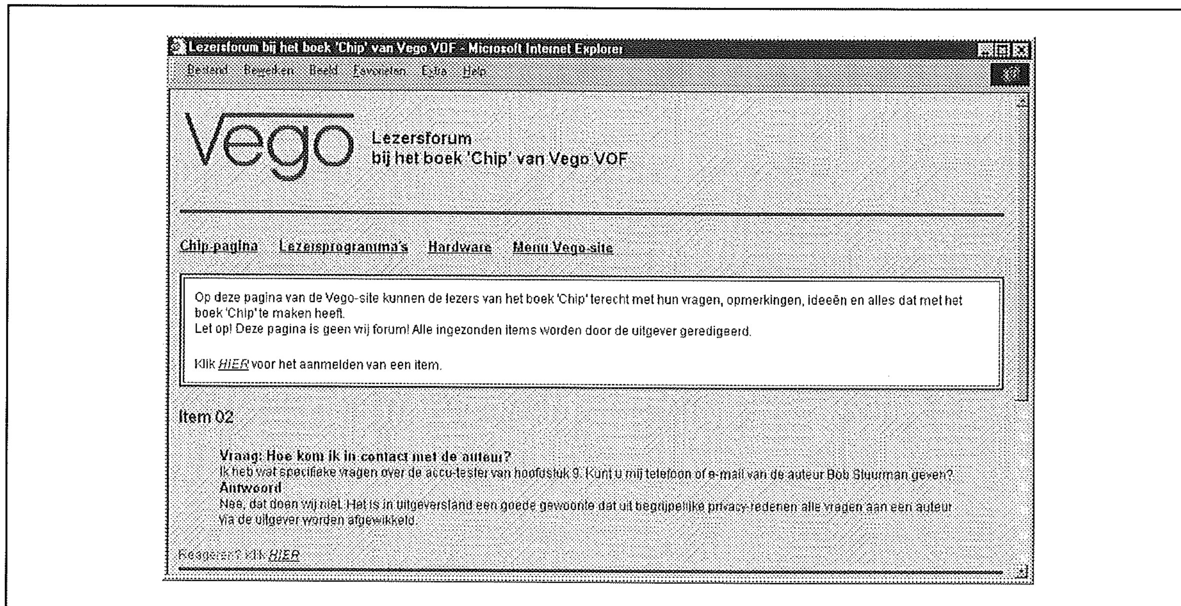
Downloaden van de software
Alle programma's die in dit en de volgende hoofdstukken zijn beschreven kunt u van de Internetpagina downloaden. De software is per hoofdstuk verzameld in een ZIP-bestand en u kunt dit ontzippen met een van de bekende ontzippers, zoals WinZip. U gaat op dezelfde manier te werk als beschreven bij het downloaden van de printen. Het enige verschil is dat u nu een ZIP-bestand op uw harde schijf krijgt en geen TIF-bestand. Dubbelklikken op het ZIP-bestand start WinZip automatisch op en u ziet onmiddellijk de bestanden die in de ZIP zitten, zie figuur 4/6.7.3-13.

Uitpakken van de software
Via de optie "Extract" van WinZip (of een soortgelijk programma) kunt u de in de ZIP aanwezige bestanden opslaan in een directory van uw harde schijf, zie figuur 4/6.7.3-14.



Figuur 4/6.7.3-14: Het uitpakken van de in het ZIP-bestand aanwezige bestanden naar een directory van uw harde schijf.

6.7 Chip, een zelfbouw computertje



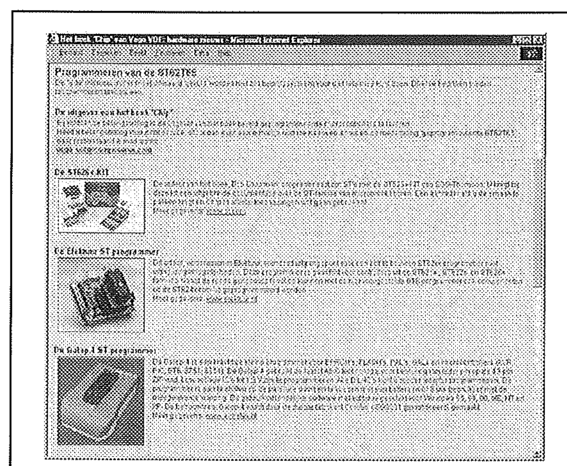
Figuur 4/6.7.3-15: De pagina "Chip-forum" wordt het podium waar u terecht kunt met al uw vragen, ideeën en opmerkingen.

De subpagina "Chip-forum"

Na het aanklikken van de link "Chip-forum" op de hoofdpagina van figuur 4/6.7.3-9 komt u terecht op een pagina die u zélf moet gaan invullen. Op deze pagina zal de uitgever uw vragen, opmerkingen, ideeën en reacties kort en krachtig samenvatten. Wij hebben uiteraard de hoop dat deze pagina uitgroeit tot een écht forum in de oorspronkelijke Romeinse betekenis van het woord. Een podium waar op een zinvolle manier van gedachten wordt gewisseld over het onderwerp van "Chip": een computertje, zo groot als twee lucifersdoosjes, waar u van alles en nog wat mee kunt.

Hardware

Via deze link op de hoofdpagina www.vego.nl/chip komt u terecht op een pagina waar allerlei nieuwtjes over de hardware van "Chip" worden verzameld. Op deze pagina treft u actuele informatie aan over adressen waar u de ST62T65 kunt kopen, hoe u deze microcontroller kunt programmeren en links naar Internetpagina's over de ST-familie van microcontrollers, zie figuur 4/6.7.3-16.



Figuur 4/6.7.3-16: Op de pagina "Hardware" treft u het laatste nieuws aan over de hardware die in dit boek wordt beschreven.

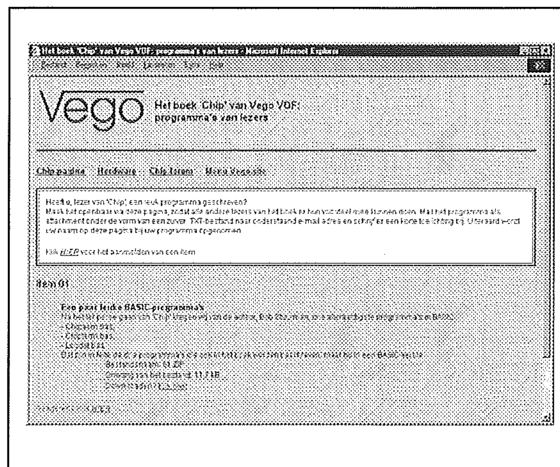
6.7 Chip, een zelfbouw computertje

Heeft u een leuke toepassing voor "Chip" bedacht? Stuur uw creatie naar de uitgever en wij nemen het op deze pagina op.

Programma's van lezers

Bob Stuurman, de ontwerper van "Chip", heeft heel wat leuke programma's voor "Chip" geschreven. Wij zullen deze in de volgende hoofdstukken publiceren. Maar uiteraard kunt u met "Chip" alle kanten op. Heeft u zélf een programma voor "Chip" geschreven? Mail het naar de uitgever en wij nemen het op deze pagina op, zie figuur 4/6.7.3-17, zodat alle lezers van dit naslagwerk er profijt van hebben.

(Bob Stuurman)



Figuur 4/6.7.3-17: Op deze "Chip"-pagina zullen wij uw eigen programma's publiceren, zodat alle lezers van "HE&IC" er gebruik van kunnen maken.

4/6.7.4

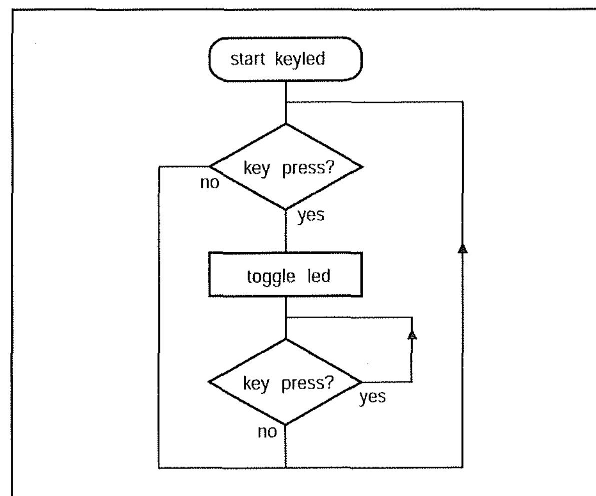
Gebruik van het keyboard en een timer

Inleiding

Bij veel programma's moet de mogelijkheid bestaan om gegevens in te voeren. Bij Chip zullen dat altijd getallen zijn, de cijfers 0 tot en met 9. Met het keyboard kan dat en de extra toetsen * en # bewijzen daarbij ook hun nut. Zo kan de *-toets bijvoorbeeld als menukeuzetoets worden gebruikt en de #-toets als bevestigingstoets. Hiermee kan dan een menusysteem worden gemaakt, bestaande uit meerdere menu's en bij de menu's behorende invoerschermen voor de ingave van de gegevens.
Het inlezen van het keyboard is dus heel belangrijk en daarom beschrijven we twee methodes, de eerste is heel eenvoudig, de tweede is wat ingewikkelder omdat het hoofdprogramma daarbij altijd doorwerkt.

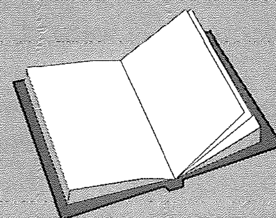
Eenvoudige keyboarduitlezing

Zolang geen toets van het keyboard is ingedrukt, is de uitleeswaarde die key geeft 3F. Zodra een toets wordt ingedrukt, wordt de waarde 30-3B, afhankelijk van de ingedrukte toets. Bij iedere volgende uitlezing wordt dezelfde waarde verkregen zolang de toets ingedrukt blijft. Maar we willen voor iedere toetsdruk slechts éénmaal de toetswaarde. Een eenvoudige methode om dat te doen is, om na inlezing en verwerking van een geldige toetswaarde, te wachten tot de toets wordt losgelaten. In figuur 4/6.7.4-1 is het stroomdiagram te zien van een op deze methode gebaseerd programma.



Figuur 4/6.7.4-1: Het stroomdiagram van de toetsenborduitlezing.

LEES OOK:
Hoofdstuk 4/6.7.1
Hoofdstuk 4/6.7.2
Hoofdstuk 4/6.7.3
www.vego.nl/chip



```
; Keyled.asm
; each keypress toggles out 0 on or off
; program waits for key release
;
start       v0 = key 0
            skip v0 <> 3f
            jp continu
;
toggle      skip out 0 = 1
            jp toggle1
            res out 0
            skip a
toggle1     set out 0
;
waitrel     v0 = key 0
            skip v0 = 3f
            jp waitrel
continu     jp start
```

Figuur 4/6.7.4-2: De listing van Keyled.asm.

Direct na de start wordt getest of er een toets ingedrukt is. Als dat het geval is, wordt een LED getoggled, dus aangezet als hij uit is en uitgezet als hij aan is. Daarna wordt gewacht tot de toets wordt losgelaten. Deze methode werkt heel goed zoals na het laden van het programma Keyled.asm (figuur 4/6.7.4-2) zal blijken. Het keyboard is aangesloten op input 0 en op output 0 is een LED aangesloten. De 3 mm (low current) LED is op een 3-polige female printheader gesoldeerd, die op output 0 is gezet. De LED kan zo dus op ieder gewenste output worden gezet, wat voor het testen van programma's handig kan zijn.

Bij iedere toetsdruk verandert de LED van toestand. Er wordt direct op iedere toetsdruk gereageerd, maar wat niet is te zien, is dat de programflow stagneert zolang de toets ingedrukt blijft door de sprong naar waitrel.

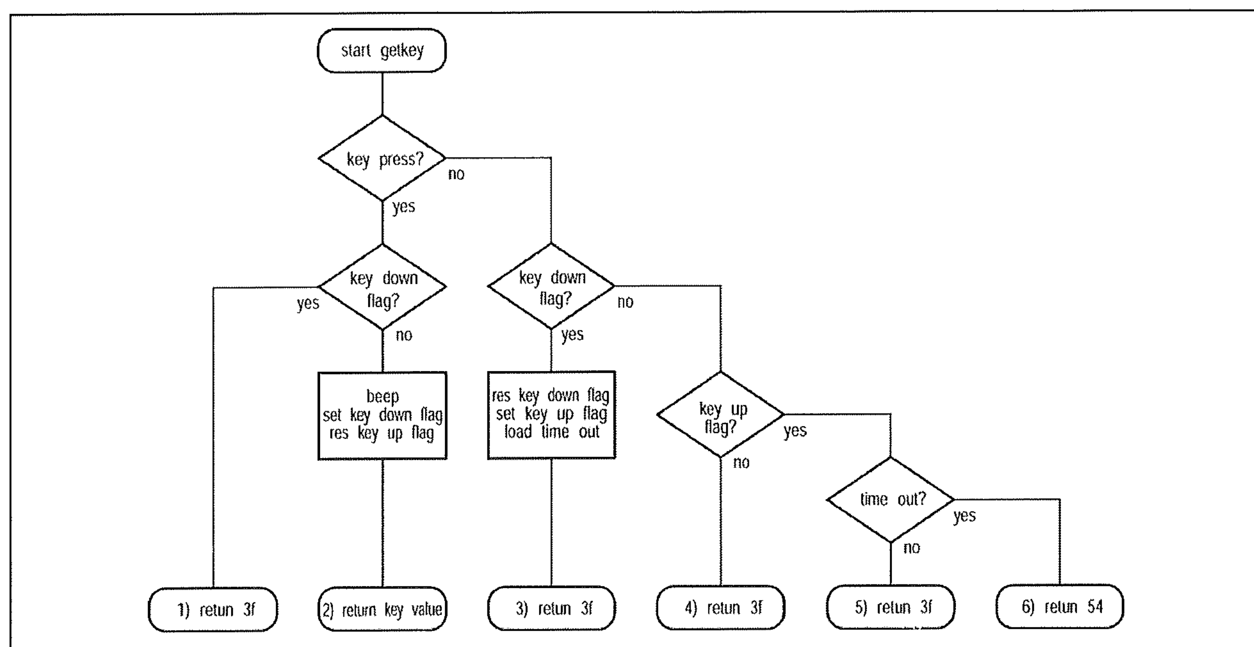
Nonstop keyboarduitlezing

Voor sommige toepassingen is het belangrijk dat een hoofdlus altijd wordt doorlopen. Dat kan bijvoorbeeld bij een acculader zijn waarbij de laadstroom op een bepaalde waarde ingesteld moet blijven. Als dan ook toetsinvoer nodig is, kan niet worden gewacht tot de toets wordt losgelaten. De uitlezing van het toetsenbord mag geen stagnatie geven in de loop van de regellus. Vandaar dat deze methode wel eens "nonstop" wordt genoemd.

In figuur 4/6.7.4-3 is het stroomdiagram van het nonstop keyboard uitleesprogramma Getkey.asm te zien. Het is een subroutine, die altijd wordt doorlopen en een "returnwaarde" geeft, die afhankelijk is van de toestand van het keyboard. Er worden twee "vlaggen" gebruikt om de toetstoestand te "onthouden". Een dergelijk gebruik van vlaggen wordt ook wel eens vergeleken met een "semafoor", een stellage waar vlaggen kunnen worden gehesen, die een bepaalde betekenis hebben.

De key down flag wordt gezet als een toets wordt ingedrukt, de key up flag als een toets wordt losgelaten. De vlaggen zijn in eerste instantie gereset (outputs van Chip).

6.7 Chip, een zelfbouw computertje



Figuur 4/6.7.4-3: Het stroomdiagram van Getkey.asm.

Uit het stroomdiagram blijkt dat de routine wordt verlaten via 4) return 3F. Na het indrukken van een toets wordt het programma via 2) verlaten met de waarde van de toets als returnwaarde. Onderweg wordt de alarmtimer geladen voor een keybeep, de key down flag gezet en de key up vlag gereset (wat bij de eerste doorgang al het geval was). Bij de volgende doorloop zal de toets nog ingedrukt zijn en wordt het programma verlaten via 1) ook met returnwaarde 3F. Na verloop van tijd wordt de toets losgelaten en bij de volgende doorloop wordt het programma verlaten via 3) met als returnwaarde weer 3F.

Nu worden onderweg de key down vlag gereset, de key up vlag gezet en een time out geladen. Als de toets los blijft en er is geen time out, dan wordt het programma via 5) verlaten met als returnwaarde 3F en als de toets niet tijdig weer wordt ingedrukt, loopt de time out af en wordt het programma via 6) verlaten met als returnwaarde 54.

De waarden 3F en 54

De waarden 3F en 54 zijn willekeurig gekozen. De returnwaarde 3F is de waarde die het keyboard geeft als geen toets is ingedrukt en is de ASCII-waarde voor ?. De waarde 54 is de ASCII-waarde voor T, toepasselijk voor time out.

De routine Getkey

Getkey moet altijd worden doorlopen en zit in een lus. Er zijn daarbij enkele zaken die aandacht vragen. In de eerste plaats moet direct na Getkey de returnwaarde worden geanalyseerd. Zolang die 3F is, is er geen geldige toets, bij 54 is er time out en een andere waarde is een geldige toets. Pas na het indrukken en weer loslaten van een toets wordt de time out gestart. De time out geldt hier dus niet voor de tijd dat een toets ingedrukt blijft. Dat is ook niet nodig, want een ingedrukte toets zal ooit wel worden losgelaten. Als het vereiste aantal toetsdrukken "binnen" is, kan de time out worden uitgezet door de key up vlag te resetten.

6.7 Chip, een zelfbouw computertje

```
; testprogram for getkey subroutine
;
; Remark: press key for some seconds, wait after key release some seconds
;
start       P = getktxt
            ld 0,f
            p = a-stack
            vd = 80               ; time out value
            ve = 03               ; beep time
;
mainlo      skip out f = 1
            jp mainlo
            res out f
            call getkey
            v0,v0 to mp
            ld f,f
            jp mainlo
;
getktxt asciz "Getkey return:  "
;
; Subroutine getkey
;
; v0 = return value:              key press           30...3b
;                                 no key press        3f
;                                 time out            54
; after first key release the time out is activated
; time out can be disabled by resetting the key up flag: out e
;
getkey      v0 = key 0
            skip v0 <> 3f         ; skip when key is pressed
            jp getkey2            ; no key press, jump
            skip out d = 0        ; skip when key down flag not set
            jp getkey1            ; key down flag is set, jump
            ve to tone            ; give beep
            set out d             ; set key down flag
            res out e             ; reset key up flag
            ret
getkey1     v0 = 3f               ; key down flag was set, return v0 = 3f
            ret
getkey2     skip out d = 1        ; skip when key down flag is set
            jp getkey3            ; key down flag not set, jump
            res out d             ; reset key down flag
            set out e             ; set key up flag
            vd to timer           ; load time out
            ret                   ; return with v0 = 3f
getkey3     skip out e = 1        ; skip when key up flag is set
            ret                   ; key up flag not set, return with v0 = 3f
            v0 = timer            ; key up flag was set, get time out
            skip v0 <> 00         ; skip on no time out
            jp getkey4
            v0 = 3f               ; no time out, return with v0 = 3f
            ret
getkey4     v0 = 54               ; time out, return with v0 = 54 ("T")
            ret
```

Figuur 4/6.7.4-4: De listing van Getkey.asm.

6.7 Chip, een zelfbouw computertje

Na de eerste toetsdruk geeft een "gezette" key up vlag aan dat de toets "los" is. In figuur 4/6.7.4-4 wordt de subroutine Getkey in een eindeloze lus door het hoofdprogramma opgeroepen. De time out waarde en de beeptijd worden vooraf insteld met vd respectievelijk ve. Deze variabelen evenals key down vlag out d en key up vlag out e moeten hier verder maar gereserveerd blijven. De time out en beep time kunnen natuurlijk ook direct worden geladen. Voor de returnwaarde wordt v0 gebruikt. Getkey wordt door het hoofdprogramma eenmaal per seconde aangeroepen, anders zou een geldige toetswaarde direct worden overschreven met 1) return 3F. Vandaar dat het keyboard ook "vertraagd" moet worden bediend.

Een timer

In de Eggtimer van figuur 4/6.7.4-5 wordt het gebruik van subroutine Getkey gedemonstreerd. Na de start van de timer verschijnt op het display een welkomboodschap met verwijzing naar de *-toets. Alleen als deze toets wordt ingedrukt, verschijnt een invoerscherm. Nu kunnen vier cijfers worden ingetoetst, twee voor het aantal minuten en twee voor het aantal seconden. De maximale waarde is 99 m 99 s. Hierbij worden alleen de cijfers 0-9 geaccepteerd. Als de laatste waarde is ingetoetst wordt bij het loslaten van de toets de timer gestart. Als de timer loopt, kunnen ook toetsen worden ingedrukt, maar alleen bij * stopt de timer en verschijnt de welkomboodschap. Het verstrijken van de tijd wordt aangegeven door een langdurige toon en de welkomboodschap.
Op output 0 kan een LED worden aangesloten, waaraan is te zien dat de hoofdlus altijd wordt doorlopen. In het programma zijn daartoe de instructies set out 0 en res out 0 opgenomen. Tijdens het verversen van het display als de timer loopt, licht de LED steeds kort wat feller op (display acties kosten vrij veel tijd), verder brandt hij op "halve kracht".
In het programma zijn vier byteposities gereserveerd voor de opslag van de invoer (inpline). Door teller vc wordt de positie op het display bijgehouden. De beginwaarde is 08, juist voor de minuten. Als door Getkey de key up flag is gezet wordt naar counter gesprongen, waar getest wordt op time out. Dan wordt de teller getest op 0Fh, in dat geval zijn de vier cijfers binnen en kunnen de minuten en secondentellers worden geladen. De key up vlag moet worden gereset en de timer run vlag worden gezet. In de hoofdlus mainloo(p) zijn direct onder elkaar twee conditionele skip opgenomen:

```
skip out c = 0
skip out f = 1
jp mainlo1
res out c
```

Alleen als de timer loopt (out c = 1) én de secondenvlag is gezet (out f = 1), wordt res out c bereikt. Res out c is nodig omdat tijdens de timerloop een toets kan worden ingedrukt waardoor de time out wordt geactiveerd.

Uitsluiten van # en *

Bij de invoer van cijfers mogen * en # niet worden geaccepteerd. Dat wordt getest door bij v0 de waarde C6 op te tellen. Een cijfer (30-39) zal geen carry geven (vf = 0).
De oorspronkelijke waarde wordt hersteld door 3A op te tellen (een carry speelt nu geen rol meer).

6.7 Chip, een zelfbouw computertje

```
; get a well boiled egg
;
start       p = intrtxt         ; show welcome text on lcd
            ld 0,f
            vc = 08             ; set character counter to 08
            vd = 80             ; time out value
            ve = 04             ; beep time
            res out c           ; reset eggtimer run flag
            res out e           ; reset key up flag
;
mainloo     set out 0           ; set led on
            skip out c = 0      ; skip when eggtimer not running
            skip out f = 1      ; skip when seconds flag set
            jp mainlo1
            res out e           ; disable time out, key could have been
pressed
            call eggshow
            skip out c = 1
            jp start            ; jump if eggtime is over
mainlo1     res out 0           ; set led off
            call getkey
            skip out e = 0
            jp counter          ; key is released, jump
            skip v0 <> 3f
            jp mainloo
            skip vc = 08
            jp number           ; counter <> 08, jump to number
            skip v0 = 3a
            jp mainloo          ; first char received <> "*"
;
gotstar     p = eggmask         ; load eggtimer mask on LCD
            ld 0,f
            vc + 01             ; set display counter to tens of minutes
            p = inpline         ; set p to input line
            res out c           ; reset eggtimer run flag
            jp mainloo
;
number      v0 + c6             ; add c6 to test for "*", "#"
            skip vf = 00
            jp mainloo          ; char > 39 ("*" or "#"), jump
            v0 + 3a             ; add 3a tot restore original value
            v0,v0 to mp         ; store number
            ld vc,vc            ; show number
            p + 1               ; increment pointer...
            vc + 01             ; ...and character counter
            skip vc <> 0b       ; if charcounter on "m"
            vc = 0d             ; set charcounter to tens of seconds
            jp mainloo
;
counter     skip v0 <> 54       ; 54 = "T"
            jp start            ; time out, jump
            skip vc = 0f        ; skip when 4 numbers have been received
            jp mainloo
;
; 4 numbers received, load the eggtimer and set eggtimer run flag
;
```

6.7 Chip, een zelfbouw computertje

```
eggload     res out e            ; reset key up flag to disable time out
            vc = 08              ; set charcounter to 08
            p = inpline          ; set pointer to inputline
            v0 = 30              ; hundreds must be set to "0"
            v1,v2 = mp           ; v1 = tens, v2 = units
            p = a-stack
            v0,v2 to mp          ; put minutes decimal number on a-stack
            v3 = 3dec mp         ; convert tot hex
            v3 to mintimer       ; load minutes timer
            p = inpline
            p + 1
            p + 1
            v1,v2 = mp           ; now get the seconds...
            p = a-stack          ; ...and repeat the same
            v0,v2 to mp
            v3 = 3dec mp
            v3 to sectimer
            set out c            ; set eggtimer run flag
            jp mainloo
;
; eggshow, routine to show minutes and seconds timer
;
eggshow     res out f
            v0 = mintimer        ; get minutes into v0
            p = a-stack
            v0 to 3dec mp        ; convert to decimal on the a-stack
            p + 1                ; point to tens
            ld 9,a               ; display tens and units
            v1 = sectimer        ; get seconds into v1
            p = a-stack
            v1 to 3dec mp        ; convert to decimal on the a-stack
            p + 1                ; point to tens
            ld d,e               ; display tens and units
            v0 or v1             ; v0 = v0 .or. v1
            skip v0 = 00         ; skip if both zero
            ret                  ; not yet zero, return
            v1 = ff              ; alert user that
            v1 to tone           ; time has run out
            res out c            ; reset eggtimer run flag
            ret
;
; Subroutine getkey
;
; v0 = return value:             key press           30...3b
;                                no key press        3f
;                                time out            54
; after first key release the time out is activated
; time out can be disabled by resetting the key up flag: out e
;
getkey      v0 = key 0
            skip v0 <> 3f        ; skip when key is pressed
            jp getkey2           ; no key press, jump
            skip out d = 0       ; skip when key down flag not set
            jp getkey1           ; key down flag is set, jump
            ve to tone           ; give beep
            set out d            ; set key down flag
```

6.7 Chip, een zelfbouw computertje

```
            res out e              ; reset key up flag
            ret
getkey1     v0 = 3f                ; key down flag was set, return v0 = 3f
            ret
getkey2     skip out d = 1         ; skip when key down flag is set
            jp getkey3             ; key down flag not set, jump
            res out d              ; reset key down flag
            set out e              ; set key up flag
            vd to timer            ; load time out
            ret                    ; return with v0 = 3f
getkey3     skip out e = 1         ; skip when key up flag is set
            ret                    ; key up flag not set, return with v0 = 3f
            v0 = timer             ; key up flag was set, get time out
            skip v0 <> 00          ; skip on no time out
            jp getkey4
            v0 = 3f                ; no time out, return with v0 = 3f
            ret
getkey4     v0 = 54                ; time out, return with v0 = 54 ("T")
            ret
;
inpline     bytes 00112233         ; storage place for received numbers
intrtxt     asciz "Eggtimer press *"; welcome text
eggmask     asciz "Eggtime: ??m ??s"; input en run time display
```

Figuur 4/6.7.4-5: De listing van Egtimer.asm.

Opmerking 1

De Eggtimer wordt "onderhuids" door de Chip realtime klok gestuurd en is daardoor op de seconde nauwkeurig.

Opmerking 2

De drie in dit hoofdstuk besproken programma's kunt u downloaden van www.vego.nl/chip, bij "Software bij hoofdstuk 4".

Bob Stuurman

4/6.7.6

Chip als robot

Opmerking

De robot die in dit hoofdstuk wordt beschreven is gebaseerd op het robotwagentje dat ook in de Basic Stamp Cursus wordt gebruikt, zie www.stampsinclass.com, www.antratek.nl en Elektuur, oktober en november 1999.

De mechanische opbouw

In het chassis zijn twee servo's gemonteerd, die eerst zijn aangepast om continu te kunnen draaien. Door ons zijn servo's FS100 gebruikt. Die kosten vrij weinig en zijn toch goed. De terugmeldingspotentiometer wordt door middel van een plastic koppelstukje in de uitgaande as aangedreven. Door dit koppelstukje te verwijderen en de blokkeringspallen van de as af te snijden wordt de servo een motor.

De stekkers van de servo's worden op de servo-uitgangen van Chip gezet.

Als achterwiel is een glad rond houten bolletje met een diameter van 25 mm gebruikt, dat is gekocht in een hobbywinkel. Bovenop, aan de voorkant van het chassis, is de Chip computer gemonteerd, met de RS232-connector naar voren wijzend. Achterop het chassis staat, op afstandbussen van 30 mm lengte, het LCD. Onder het chassis, tussen de servo's, hangt het accupakket (vier NiCad penlites in een houder) dat met een kabelbandje vastzit. Het accupakket is zo geplaatst dat de druk op het "achterwiel" niet hoog is en dit gemakkelijk zijwaarts kan schuiven. Tegen de achter zijkant van de robot is, met dubbelzijdig plakband, de aan/uit-schakelaar bevestigd.

De mechanische opbouw van de robot is voorgesteld in figuur 4/6.7.6-1.

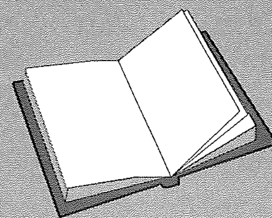
Obstakel detectie

Voor de detectie van obstakels is aan de voorkant van het chassis een bumperschakelaar gemonteerd en voorop het chassis zijn links en rechts infraroodsensors (zie verder) opgenomen.

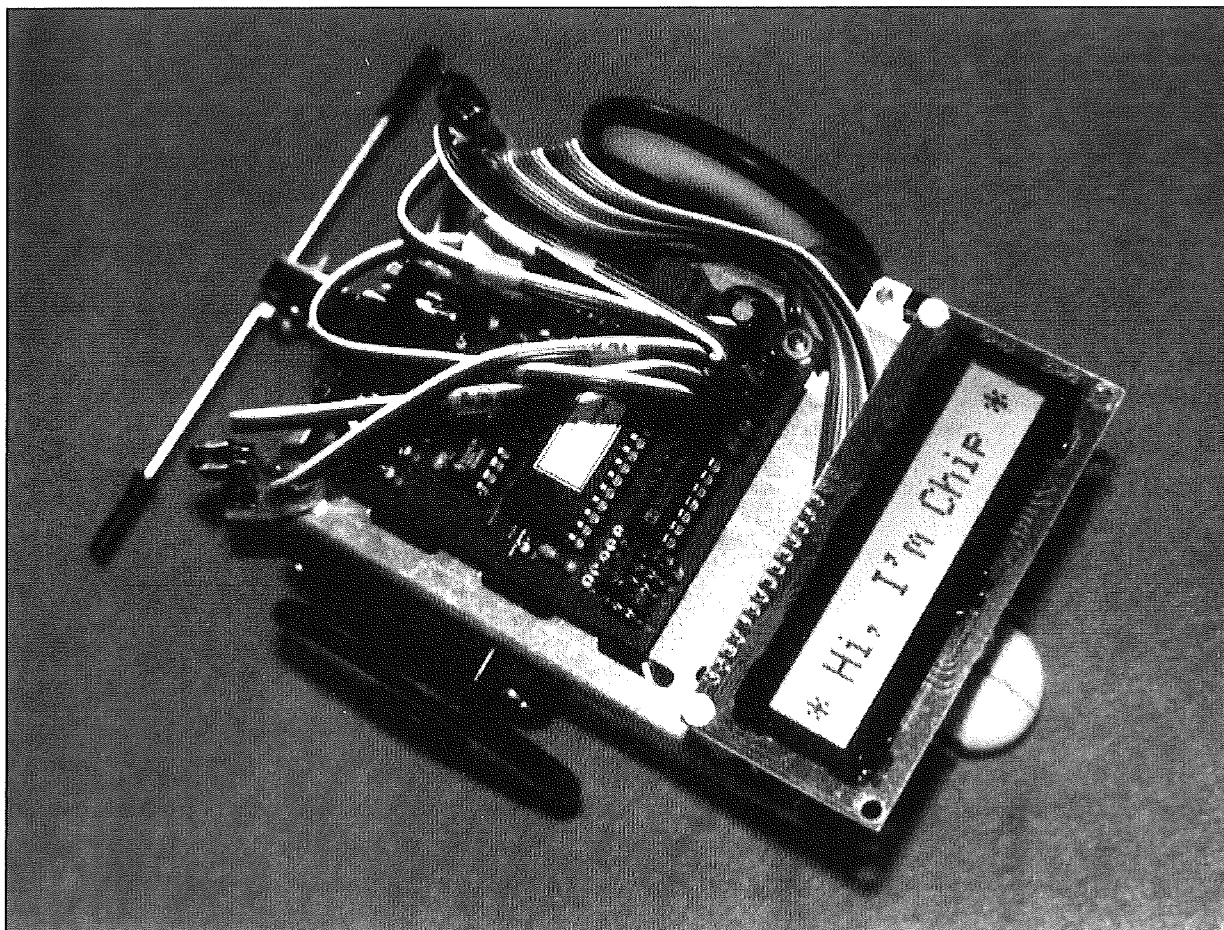
Voor de bumperschakelaar is een microswitch gebruikt met aan de bedieningshevel een stuk messing buis (diameter 3 mm, lengte 100 mm). Over de einden zijn stukjes rubberslang geschoven, die aan beide zijden ongeveer 1 cm

LEES OOK:

Hoofdstuk 4/6.7.1
Hoofdstuk 4/6.7.2
Hoofdstuk 4/6.7.3
Hoofdstuk 4/6.7.4
www.vego.nl/chip



6.7 Chip, een zelfbouw computertje



Figuur 4/6.7.6-1: De robot is opgebouwd uit standaard onderdelen.

buiten het wagentje uitsteken. De "bumper" zit 20 mm boven de grond en 40 mm voor de robot voorkant. Door de constructie is de bumper niet star, bij een botsing geeft hij mee. De switch is met een twee-aderig snoer met een ingang verbonden, tussen in en Gnd.

Neutral.asm

Met behulp van het programma Neutral.asm, voorgesteld in de listing van figuur 4/6.7.6-2, kunnen de servo's neutraal worden gesteld. De bytes op adres 000C bepalen de draaisnelheid van servo 1 respectievelijk servo 2. Een waarde van 80 is de standaardwaarde (neutraal). Het instellen moet vrij nauwkeurig gebeuren.

De infrarood sensors

De infrarood sensors zijn gebaseerd op het "Lego Robotics Invention System" en het schema ervan is getekend in figuur 4/6.7.6-3. De sensor IS471F bevat een oscillator voor de sturing van de infrarood LED.

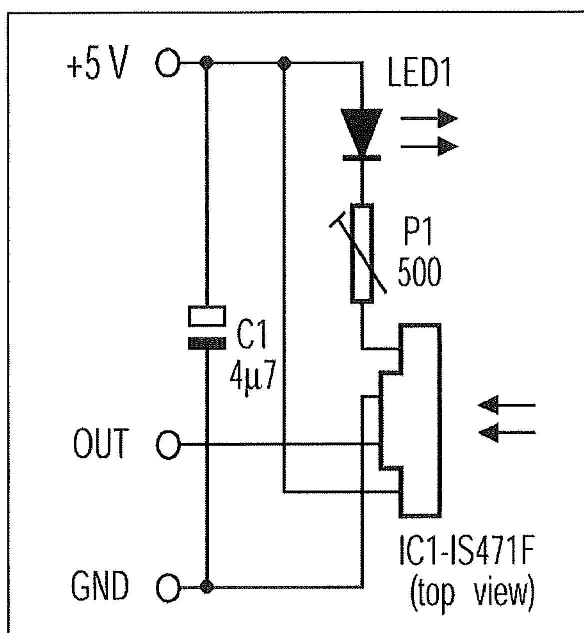
De bouw van de sensor

Figuur 4/6.7.6-4, op de laatste pagina van dit hoofdstuk, laat de print zien en figuur 4/6.7.6-5 de opstelling van de onderdelen.

6.7 Chip, een zelfbouw computertje

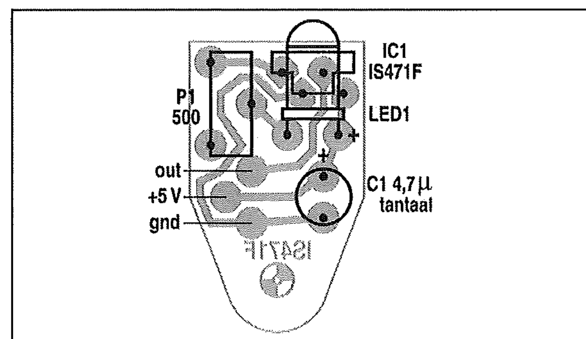
```
; Listing Neutral.asm, aid in adjusting servos
;
neutral son
        p = neutral     ; point to neutral string byte
        v0, v1 = mp     ; copy into v0, v1
        v0 to s1        ; copy v0 into servo 1
        v1 to s2        ; and v1 into servo 2
        break           ; return to command mode
neutral bytes 8080
```

Figuur 4/6.7.6-2: De listing van Neutral.asm.



Figuur 4/6.7.6-3: Het schema van één infrarood sensor.

De infrarood LED zit boven de sensor. Tussen de LED en de sensor zit een stukje zwart papier, anders "ziet" de sensor de LED. Aan het einde van het sensorsnoer is een driepolige female printheader gesoldeerd, met over de verbindingen krimpkousjes. Deze past op de ingangen van Chip. De printheader is met een figuurzaagje voor metaal van de strip afgezaagd. De sensors zijn niet zo gevoelig: zo'n 15 tot 20 cm, maar dat is hier juist voldoende.



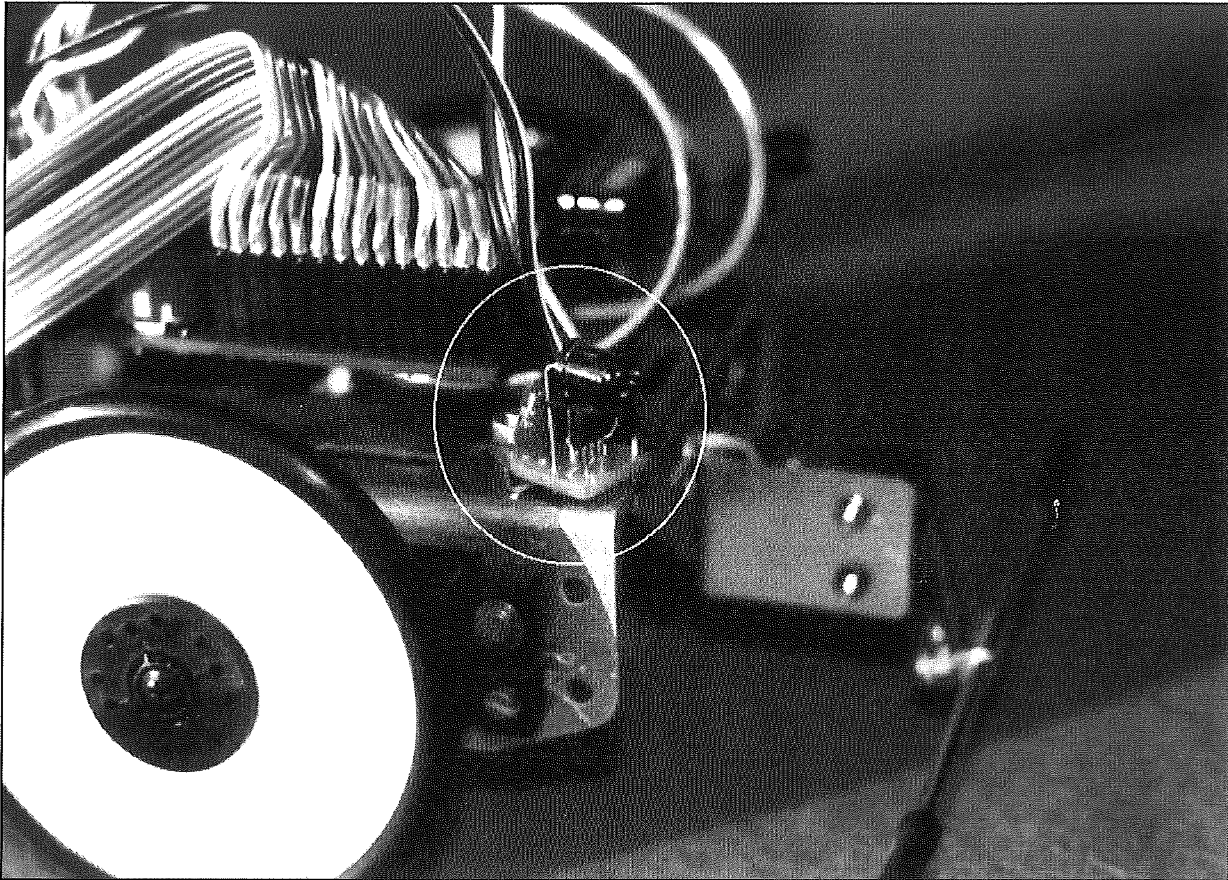
Figuur 4/6.7.6-5: De componentenopstelling van de sensorprint.

De montage van de sensoren
Op de foto van figuur 4/6.7.6-6 is te zien hoe een sensorprintje op het chassis van de robot wordt bevestigd.

ONDERDELENLIJST INFRAROOD SENSOR

IC1	IS471F, Conrad best. nr. 185094
LED1	SFH409, Conrad best. nr. 183776
P1	Piher instelpotentiometer 500 Ω, 6 mm staand
C1	4,7 µF, 35 V tantaal, RM 2,54
1	3-aderig snoer
1	3-pens female SIL printheader

6.7 Chip, een zelfbouw computertje



Figuur 4/6.7.6-6: Het bevestigen van de sensorprintjes op het chassis van de robot.

Het robotprogramma Robot.asm

De listing in figuur 4/6.7.6-7 is het volledige robotprogramma Robot.asm voor Chip. Na assemblage kan het in de EEPROM worden geladen, die is dan voor circa 20 % gevuld. Het programma wordt gestart door de schakelaar aan te zetten, terwijl de drukknop S1 is ingedrukt of de autostart jumper te plaatsen. De robot begint niet direct te rijden, eerst is er de "count down" van tien seconden. Dit is op het LCD te zien. Dan gaat de robot rijden.
De snelheid en richting worden door middel van toevalsgetallen uit een tabel gehaald. In eerste instantie werd de richting volkomen willekeurig gekozen, maar dat was niet leuk. Het is veel spannender als de robot enigszins zwalkend vooruit rijdt. Het lijkt dan, of hij tegen een obstakel aan zal botsen, maar meestal rijdt hij er net langs en soms botst hij er tegen op. Als hij er tegen op botst (bumper collision), gaat hij eerst een stukje achteruit en draait dan links- of rechtsom. Dit hangt weer af van een toevalsgetal, nadien begint hij weer vooruit te rijden.

Obstakels

Als een obstakel wordt gezien door een IR-sensor, dan gaat de robot een stukje achteruit, terwijl hij iets weg draait van het obstakel en dan iets vooruit, eveneens wegdraaiend van het obstakel, waarna hij weer zwalkend vooruit gaat. Als di-

6.7 Chip, een zelfbouw computertje

rect, nadat een IR-sensor een obstakel heeft gezien, de andere IR-sensor een obstakel ziet, dan wordt dit afgehandeld als een bumperbotsing. Terwijl de robot rond rijdt worden op het display mededelingen gezet over de stand van zaken. Als obstakels worden herkend piept de robot. In de listing is te zien, dat veelvuldig gebruik wordt gemaakt van toevalsgetallen en dat het merendeel van de te gebruiken gegevens in tabellen staat. Door middel van de pointer kunnen deze waarden snel in variabelen worden geladen.

Rijgedrag
Het is werkelijk geen vertoning zoals de Chip-robot door de kamer rijdt. Iedereen die het ziet is met stomheid geslagen. Hij rijdt weliswaar niet zo snel, maar het is de onvoorspelbaarheid van zijn rijden, dat het spannend maakt. Soms zit hij in een hoek (een verhuisdoos is heel leuk) en denk je "daar komt t'ie nooit uit", en het volgende ogenblik rijdt hij alweer rustig zwalkend rond. Als hij vast zit, komt dat bijna altijd doordat de bumper komt klem te zitten, bijvoorbeeld tussen twee stoelpoten. Tegen dergelijke hinderlagen is Chip niet opgewassen.

```
; Listing Robot.asm
; a robot program for Chip
;
; The robot mechanics are based on the robot from parallax.
; The infrared sensors are based on the IS471F integrated
; circuit.
;
; connections:          left servo = s1
;                       right servo = s2
;                       infra red detector left = in 0
;                       infra red detector right = in 1
;                       bumper switch = in 2
;
; note 1: detectors are normally high
; note 2: output f is set every second by the operating system
; note 3: do not use v0 for a random number, conflicts with
; 00nn instructions
; note 4: vb is used to get robot out of a corner, when there
; is an obstacle
; right or left, immediately followed by an obstacle on the
; other side, this is handled as a collision.
;
start   p = contxt      ; point p to count down text
        ld 0,f          ; and load display
        v0 = 0a         ; load v0 with count down delay
        v0 to sectimer ; put into seconds timer
        set out e       ; set flag e for counting while
                        ; counting down
start1  skip out e = 1 ; skip jump to main while flag e = set
        jp main
```

6.7 Chip, een zelfbouw computertje

```
        skip out f = 0 ; skip call while flag f = 0
        call countdn   ; flag f = set by opsys, time to call
                       ; sub
        jp start1      ; and loop while flag e
;
; count down subroutine
;
countdn res out f      ; reset calling flag f
        p = a-stack    ; point p to a-stack for conversion
        v0 = sectimer  ; get seconds into v0
        v0 to 3dec mp  ; convert v0 to decimal on a-stack
        p + 1          ; point to tens
        ld a,b         ; load display with tens and units
        skip v0 <> 00  ; skip resetting flag e while
                       ;seconds <> 00
        res out e      ; seconds = 00, reset flag e
        ret            ; return
;
main    son            ; enable servo drive
        p = wantxt     ; point p to wandering text
        rotate         ; rotate text on display
main1   v0 = timer     ; get the timer into v0
        skip v0 <> 00  ; skip jump to wander while v0 (timer)
                       ; <> 00
        jp wander
main2   skip in 2 = 1  ; skip if no bumper collision
        jp collisi     ; yes, collision, jump to handling
                       ; routine
        call irsense   ; read infra red sensors
        skip va <> 00  ; if va <> 00, there was an obstacle
        jp main1       ; no obstacle, loop
        skip va <> 01
        jp obsleft     ; va = 01, obstacle left
        skip va <> 02
        jp obsrigt     ; va = 02, obstacle right
        jp obsfron     ; va must be 03, obstacle front
;
wander  v1 = rnd,1f    ; v1 = random 0 to 1f
        v1 + 10        ; v1 = random 10 to 2f
        v1 to timer    ; load timer with random number v1
        v1 = rnd,07    ; v1 = random 0 to 7
        v1 to v0       ; copy v1 into v0
        v1 + v0        ; v1 = random number * 2
        p = tablew     ; point p to start of direction table
        p + v1         ; add v1 for random table entry
        v0, v1 = mp    ; get table values into v0 to v1
        v0 to s1       ; load servo drives
        v1 to s2
```

6.7 Chip, een zelfbouw computertje

```
        vb = 00         ; reset obstacle right/left v.v flag
        jp main2        ; and jump back
;
collisi call stopser    ; stop servo's
        stop rotate     ; stop text rotate, clear display
        p = coltxt      ; and show collision text
        ld 0,f
collis1 p = esccoll     ; point p to start of collision table
        v1 = rnd, 08    ; v1 = a random number 0 or 8
        p + v1          ; p points to esccoll or esccolr
        call escape     ; execute the instructions
        vb = 00         ; reset obstacle right/left v.v flag
        jp main1        ; and jump back
obsleft call stopser    ; stop servo's
        stop rotate     ; stop text rotate, clear display
        p = leftxt      ; and show obstacle left text
        ld 0,f
        skip vb <> 03   ; skip if .not. obstacle right/left
                        ; or v.v.
        jp collisi
        p = escleft     ; point p to escape left table
        call escape     ; execute the instructions
        jp main1        ; and jump back
obsrigt call stopser    ; stop servo's
        stop rotate     ; stop text rotate, clear display
        p = rigtxt      ; and show obstacle right text
        ld 0,f
        skip vb <> 03   ; skip if .not. obstacle right/left
                        ; or v.v.
        jp collisi
        p = escrigt     ; point p to escape right table
        call escape     ; execute the instructions
        jp main1        ; and jump back
obsfron call stopser    ; stop servo's
        stop rotate     ; stop text rotate, clear display
        p = frotxt      ; and show obstacle front text
        ld 0,f
        jp collis1      ; let collision handle this
;
; stop servo's subroutine
;
stopser p = stop        ; point p to stop servo values
        v0, v1 = mp     ; load values into v0 to v1
        v0 to s1        ; load the servo's
        v1 to s2
        v0 = 06
        v0 to tone      ; give a short beep
        ret
```

6.7 Chip, een zelfbouw computertje

```
;
; infra red sensors subroutine
;
irsense va = 00          ; preset result value to 00
        v0 = 01          ; v0 is .or. mask left
        v1 = 02          ; v1 is .or. mask right
        skip in 0 = 1    ; skip no obstacle left
        va or v0         ; obstacle, .or into result
        skip in 1 = 1    ; skip no obstacle right
        va or v1         ; obstacle, .or. into result
        skip in 0 = 1    ; skip no obstacle left
        va or v0         ; obstacle, .or. into result
        vb or va         ; save result in vb
        ret
;
; escape subroutine
;
escape  v0, v5 = mp      ; load variables v0 to v5 from mp
        v0 to s1         ; v0 goes to servo 1
        v1 to s2         ; v1 goes to servo 2
        v2 to timer      ; v2 goes to the timer
escape1 v2 = timer       ; wait till timer = 00
        skip v2 = 00
        jp escape1
        v3 to s1         ; v3 goes to servo 1
        v4 to s2         ; v4 goes to servo 2
        v5 to timer      ; v5 goes to the timer
escape2 v5 = timer       ; wait till timer = 00
        skip v5 = 00
        jp escape2
        call stopser     ; stop the servo's
        p = wantxt       ; point p to wantxt
        rotate           ; load display
        p = tablew       ; set course straight forward
        v0, v1 = mp
        v2 = 10          ; for appr. 0.6 seconds
        v2 to timer      ; necessary to detect left obstacle
        v0 to s1         ; immediately after right obstacle or
                         ; vice versa
        v1 to s2         ; load servo drives
        ret              ; and return
;
; the random direction table for wandering
; note the duration is also set at random
;
tablew  bytes 778a       ; straight forward
        bytes 768a       ; increase speed left wheel
        bytes 758a       ; increase speed left wheel
```

6.7 Chip, een zelfbouw computertje

```
        bytes 748a     ; increase speed left wheel
        bytes 778a     ; straight forward
        bytes 778b     ; increase speed right wheel
        bytes 778c     ; increase speed right wheel
        bytes 778d     ; increase speed right wheel
;
stop    bytes 8080
;
contxt  asciz "countdown    sec"
wantxt  asciz "Inspecting premises"
coltxt  asciz "Bumper Collision"
leftxt  asciz "Obstacle left!  "
rigtxt  asciz "Obstacle right! "
frotxt  asciz "Obstacle front! "
;
; the table for collision and obstacles
; note: esccoll and esccolr are selected at random
;
esccoll bytes 847c20888835   ; go back, rotate left
        bytes 0000           ; fillers to get collision table
                             ; offset of 8
esccolr bytes 847c20797930   ; go back, rotate right
escleft bytes 80790f797910   ; go right back, turn left
                             ; a little
escrigt bytes 87800f878710   ; go left back, turn right
                             ; a little
;
; end of robot
```

Figuur 4/6.7.6-7: De listing Robot.asm.

(Bob Stuurman)

De listings uit dit hoofdstuk kunt u downloaden van www.vego.nl/chip

6.7 Chip, een zelfbouw computertje

6.7 Chip, een zelfbouw computertje



Figuur 4/6.7.5-4: De print voor de sensor.

HOE MAAKT U DEZE PRINT?

OPTIE 1: zelf maken

U scant deze pagina en drukt deze met een inkjet-printer af op A4 formaat op transparante folie. U knipt de print uit en belicht er de fotogevoelige printplaat mee.

OPTIE 2: via Internet

Op **www.hobbyelektronica.nu** selecteert u uit het linker menu de optie "Printservice". In het rechter venster selecteert u het hoofdstuknummer. U kunt nu de print als TIF-file downloaden. U opent deze file in een beeldbewerkingsprogramma en drukt deze met de op de Internet-pagina aangegeven afmetingen op transparante folie af. U belicht hiermee de fotogevoelige print.

OPTIE 3: bestellen

U stuurt een **ONGEFRANKEERD** briefje naar Vego VOF, Antwoordnummer 30020, 6374 ED Landgraaf, met vermelding van het hoofdstuknummer. U krijgt per kerende post het printontwerpje op transparante folie **GRATIS** toegestuurd. U belicht hiermee de fotogevoelige print.

6.7 Chip, een zelfbouw computertje

4/6.7.7

Chip als klok

Inleiding

Als je met plezier aan het werk bent, vliegt de tijd voorbij. Een klok waarop je kunt zien of het tijd voor koffie is, is onmisbaar. Chip kan heel goed als klok worden gebruikt, maar om hem alleen daarvoor te gebruiken is onterecht, daarvoor is hij te "knap". Omdat Chip weet welke week en welke dag het is, kan hij ook als verjaardagskalender worden gebruikt en om de puntjes op de i te zetten laten we hem tikken en geven hem een slagwerk waarbij, voor het slaan van de tijd, een melodietje is te horen.

Een klok

De realtime klok van Chip wordt gestuurd door de interrupt routine van timer 1. De klok werkt dus "op de achtergrond" en zo lang we Chip niet uitschakelen blijft hij precies op tijd lopen, mits het afregelbyte correct is ingesteld. De klok houdt de weken, dagen, uren, minuten en seconden (zie de tabel van figuur 4/6.7.7-1) bij en is dus uitermate geschikt voor homesystemen omdat het dagnummer aangeeft of het een doordeweekse- of een weekenddag is.

Clock.asm

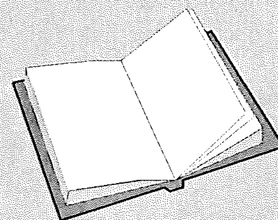
Clock.asm, zie de listing van figuur 4/6.7.7-2, laat de subroutine zien die de tijd op het display zet. Het aardige aan deze routine is dat gebruik wordt gemaakt van de variabelen v0 en v1 om de posities op het display te adresseren.

AAh	weken
ABh	dagen
ACh	uren
ADh	minuten
AEh	seconden

Figuur 4/6.7.7-1: Chip's inwendige tijdregisters.

Hierdoor kunnen we een lus maken en zijn de conversie- en display-instructie slechts éénmaal nodig. Een nadeel is dat de routine wat meer tijd in beslag neemt. De klok wordt eenmaal per seconde op het display gezet, vaker is immers niet nodig. Het spaart processortijd en maakt het display rustiger om te zien.

LEES OOK:
Hoofdstuk 4/6.7.1
Hoofdstuk 4/6.7.2
Hoofdstuk 4/6.7.3
Hoofdstuk 4/6.7.4
www.vego.nl/chip



```
; Listing Clock.asm
; Clock, subroutine to show the week, day of week and time on the LCD.
; Note that we use variables to load the display,
; so we can use a loop.
;
main     p= clock2        ; point to the initial clock text
         ld 0,f           ; and load the initial display
main0    skip out f = 0   ; skip if the seconds flag is not set
         call clock       ; call the clock sub
         jp main0         ; loop
;
         org 20           ; let the clocksubroutine begin here
clock    res out f        ; reset the seconds flag
         v0 = 01          ; initialise v0 and v1
         v1 = 02
clock1   skip v0 <> 01    ; get the corresponding byte
         v3 = weeks       ; and load into v3
         skip v0 <> 04
         v3 = days
         skip v0 <> 07
         v3 = hours
         skip v0 <> 0a
         v3 = minutes
         skip v0 <> 0d
         v3 = seconds
         p = a-stack      ; we let p point to a-stack
         v3 to 3dec       ; and convert v3 to 3 decimals
         p+1              ; we don't need the hundreds
         ld v0,v1         ; show tens and units on display
         v0 + 03          ; let v0 and v1 point
         v1 + 03          ; to next display position
         skip v0 = 10     ; if v0 = 10 we are ready
         jp clock1        ; else we loop
         ret
clock2   asciz "   :  :  :  :   "; this is the initital display
```

Figuur 4/6.7.7-2: De listing van Clock.asm

De klok kan worden ingesteld met het commando time, waarbij het aan de gebruiker wordt overgelaten om voor de zondag, dag zeven of dag één te kiezen.

Chip leert klok kijken

Wij kijken op de klok en weten dan of we naar de tandarts moeten of nog niet. Chip moet ook kunnen "klok kijken" om te weten of "er iets moet gebeuren". Met het programma Dattime.asm in de listing van figuur 4/6.7.7-3 kan Chip klok kijken. De "gebeur-tijd" staat in de bytesstring datstr1, in dit geval week 41, dag 06, 09 uur, 15 minuten en 00 seconden precies. In de hoofdroutine wordt de pointer op de string gezet en de subroutine wordt aangeroepen.

6.7 Chip, een zelfbouw computertje

```
; Listing Dattime.asm
;
main     p = datstr1
main1    set out 4
         call dattime
         res out 4
         skip vf = 00
         jp sound
         jp main1
sound    v0 = 80
         v0 to tone
         jp main1
;
datstr1  bytes 2906      ; weeks, days
         bytes 090f      ; hours, minutes
         bytes 00        ; seconds
;
; dattime, subroutine to compare the running clock against a
; date-time string
; pointed to by the pointer:
; p -> weeks, days, hours, minutes, seconds (all values hex!)
; when a dattim string byte is 7f, it will not be tested
; when the condition is met for the first time, vf will be set as
; flag (<> 00)
; and bit 7 of weeks, date-time string will be set. As soon as
; the condition
; is no more valid this bit wil be reset.
;
dattime  v0,v4 = mp      ; move string date and time into v0...v4
         vf = 7f         ; strip bit 7 from v0
         v0 and vf
         skip v0 <> 7f   ; skip if weeks <> 7f
         jp dattim1      ; weeks = 7f, so do not test
         vf = weeks      ; get clock weeks
         skip v0 = vf    ; skip if equal
         jp dattim7      ; not equal, jump
dattim1  skip v1 <> 7f   ; skip if days <> 7f
         jp dattim2      ; days = 7f, so do not test
         vf = days       ; get clock days
         skip v1 = vf    ; skip if equel
         jp dattim7      ; not equal, jump
dattim2  skip v2 <> 7f   ; skip if hours <> 7f
         jp dattim3      ; hours = 7f, so do not test
         vf = hours      ; get clock hours
         skip v2 = vf    ; skip if equal
         jp dattim7      ; not equal, jump
dattim3  skip v3 <> 7f   ; skip if minutes <> 7f
         jp dattim4      ; minutes = 7f, so do not test
```

6.7 Chip, een zelfbouw computertje

```
        vf = minutes    ; get clock minutes
        skip v3 = vf    ; skip if equal
        jp dattim7      ; not equal, jump
dattim4 skip v4 <> 7f   ; skip if seconds <> 7f
        jp dattim5      ; seconds = 7f, so do not test
        vf = seconds    ; get clock seconds
        skip v4 = vf    ; skip if equal
        jp dattim7      ; not equal, jump
;
; dattim string is equal to the clock
; when this happens for the first time, bit 7 of v0 (weeks) will be 0,
; it must be set to 1 and vf must be made <> 00
;
dattim5 v0, v0 = mp
        vf = 80         ; vf is bitmask
        vf and v0       ; strip d6...d0
        skip vf = 00    ; skip if d7 = 0
        jp dattim6      ; jump, this dattim skip has alread been taken
        vf = 80         ; vf is .or. bit 7
        v0 or vf        ; set 7 of v0
        v0, v0 to mp    ; write back to dattim string
        skip a          ; skip always, let vf remain 80
dattim6 vf = 00         ; entry point for skip already taken,
                        ; reset flag
        ret
dattim7 v0, v0 to mp    ; write back to dattim string
        vf = 00         ; reset flag vf
        ret
```

Figuur 4/6.7.7-3: De listing van Dattime.asm.

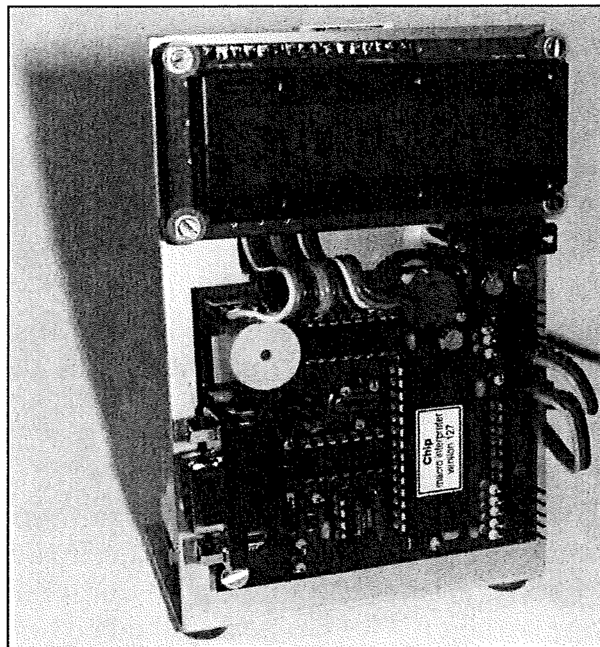
Als de klok gelijk is aan de “stringtijd” wordt vf <> 00 gemaakt waarop de roepende routine actie kan ondernemen. In het voorbeeld wordt een geluidssignaal gegeven.
De Dattime subroutine laadt eerst de stringbytes in variabelen waarbij bit 7 van de weken wordt gestript. Vervolgens worden de variabelen vergeleken met de klok, tenminste als ze ongelijk 7Fh zijn. Als alles gelijk is, en in de tijdstring bit 7 van de weken 0 is, wordt vf gezet (<> 00) en het genoemde bit wordt gezet. Zodra de tijden ongelijk zijn geworden, wordt dit bit weer gestript. Een 7Fh byte is een Joker, een “doet er niet toe” byte. Als we deze waarde als week invullen, wordt de string iedere week geldig en als we voor weken en dagen 7Fh invullen, iedere dag. Als we voor de seconden 7Fh invullen, hebben we een minuut de tijd voor de string “ongeldig” wordt. Voor de test op een weekend gebruiken we natuurlijk de instructie vx = days en testen vx. Er kunnen meerdere tijdstrings in een programma worden opgenomen en de string waarop de pointer staat wordt getest. In het voorbeeld is de verjaardag van auteur dezes opgenomen, verander dat maar in uw eigen verjaardag of die van een gewaardeerd iemand. De instructies set out 4 en res out 4 zijn opge-

6.7 Chip, een zelfbouw computertje

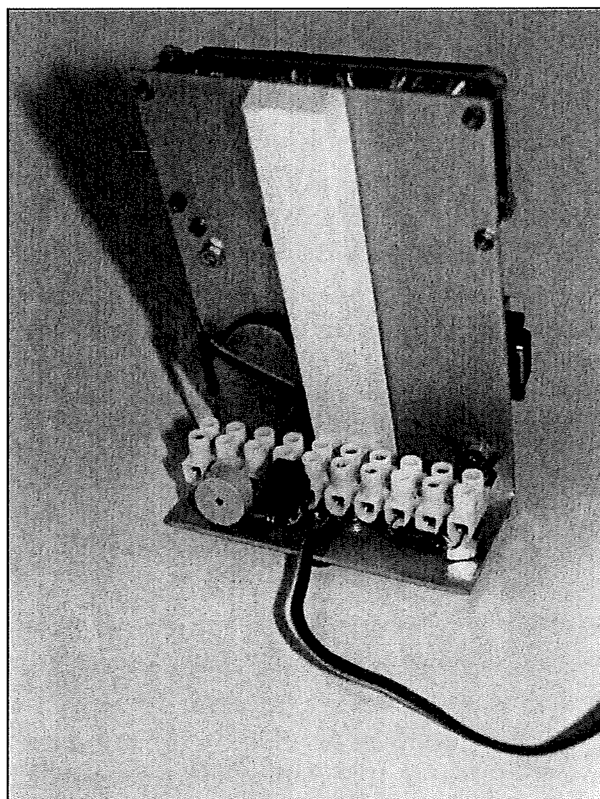
nomen om met de scoop de tijden de kunnen meten.

Een hi-tech klok
Onze wekker was kapot. We waren eraan gehecht want ruim 35 jaar heeft hij trouw voor ons getikt. Het was een van de eerste wekkers waarbij de onrust via een transistor elektromagnetisch werd aangedreven. De rest was zuiver mechanisch. Onze gedachten gingen naar Chip. Deze bevat alle componenten voor een aardig klokje en ook voor wat betreft de software waren er interessante mogelijkheden. Van een plaatje aluminium 160 mm hoog, 82 mm breed, dik 1,5 mm hebben we aan de onderkant een strook van 40 mm omgezet, iets meer dan 90 graden, zie figuur 4/6.7.7-4. De omgezette strook is de bodem. Boven aan de voorkant hebben we het display gemonteerd, met daaronder Chip. Tussen het display en Chip zit de aan/uit- schakelaar. Weliswaar gebruikt Chip slechts 8 mA, maar dat is teveel voor een klokje met batterijvoeding. Daarom is een (ongestabiliseerd) netstekkervoedinkje gebruikt dat 300 mA kan leveren en dat omschakelbaar is tussen 3 V, 4,5 V, 6 V, 7,5 V, 9 V en 12 V. Voor de voeding van Chip is een low drop 5 V stabilisator gebruikt. Omdat het makkelijk is als het klokje ook in het donker afleesbaar is, hebben we een display met backlight gebruikt dat via een serieweerstand van 27 Ω, 2 W op de voeding is aangesloten. Met de spanningkeuze kan het backlight tot heel helder worden ingesteld.

Kroonstrip voor de aansluitingen
Aan de achterkant van de constructie hebben wij een kroonstrip gemonteerd waarop alle externe bedrading kan worden aangesloten, zie figuur 4/6.7.7-5.



Figuur 4/6.7.7-4: Het high tech klokje waarschuwt bij verjaardagen.



Figuur 4/6.7.7-5: Handig, de kroonsteenstrip aan de achterkant voor montage van onderdelen.

6.7 Chip, een zelfbouw computertje

Opmerking
Bij 12 V wordt de weerstand vrij warm, hij dissipeert dan ongeveer 1,5 W.

Tikken en melodietje
Onze wekker tikte, dus het nieuwe klokje moest ook tikken. Net als een echte klok moest hij ook bij de hele uren slaan en om het geluidenpallet compleet te maken moest hij, voor dat "slaan van de uren", een muziekje spelen.

Chip onthoudt verjaardagen
Toch ontbrak er nog iets. Weliswaar hebben we een verjaardagskalender, maar we kijken daar te weinig op en vergeten verjaardagen. Het klokje kon daarbij helpen, want Chip kan immers "klok kijken". Zo hebben we nu een heel bijzonder klokje, dat voor een technisch iemand een plezier is om naar te kijken, maar dat ook wat betreft functionaliteit zijn weerga niet kent.
Om de muziek te laten horen moet op servo 1 een piëzo sounder worden aangesloten, eenzelfde type als op de Chip print zit. En om een roterende felicitatiewens te kunnen stoppen, moet op input 0 (en Gnd) een drukknop worden aangesloten.

Calclock.asm
Het klokprogramma Calclock.asm (Calendar and clock) is te zien in de listing van figuur 4/6.7.7-6. Eerst wordt de PWM-timer omgeprogrammeerd om muziek te spelen. In de hoofdlus mainlo(op) worden vlag 5 en vlag 6 getest. Als vlag 5 is gezet wordt een muziekje (Chimes) gespeeld en als vlag 6 is gezet wordt het aantal uren geslagen (Hourwrk). Er zijn vier liedjes, via een toevalsgetal (random number) wordt er een gekozen. Omdat de Chip 24 uren aangeeft, wordt vanaf 13:00 uur een correctie aangebracht en bij 00:00 uur (12 uur 's nachts) wordt twaalf maal geslagen. Eenmaal per seconde wordt de tijd getoond door subroutine Clock. De subroutines Clock, Chimes en Hourwrk sturen de vlaggen. Aardig is dat bij het slaan van de uren de clock doorloopt.
De verjaardagen worden niet alle iedere seconde getest, maar per seconde één, te beginnen bij seconde 45. Eerst wordt de pointer op datsent gezet, dan wordt de pointer op een (bij een seconde horende) dattim string gezet. Als de pointer niet is veranderd, is dat de zien aan de byte 00 waarop hij wijst en gaan we terug naar mainloop.
Als hij wel op een dattim string staat, zal subroutine dattime vf <> 00 maken bij een verjaardag op de ingestelde tijd. Precies zes 6 posities lager staat de felicitatieboodschap die gaat roteren op het display. Het roteren stopt als input 0 wordt laag gemaakt of als het 10:00 uur wordt. Practisch gezien worden alle tijdstrings eenmaal per minuut getest. Daarom zijn in de strings de seconden op "don't care" (7fh) ingesteld. Anders zouden we veel boze gezichten krijgen. Chip werkt met weken en dagen, daarom moeten de verjaardagen worden omgezet in deze notatie en natuurlijk in hex. Aan het begin van ieder jaar moeten de voor dat jaar geldende strings worden geladen, maar dat betreft dan alleen het deel beginnende met org 400.
In de dattim-strings staat een tijd van 07:45 uur. Dat is de tijd waarop bij een verjaardag de gelukwens verschijnt. Misschien heeft u liever dat dit op een heel uur gebeurt. Dat kan natuurlijk, vandaar ook dat de tests pas beginnen vanaf seconde 45, dan is het slagwerk al lang afgelopen.

6.7 Chip, een zelfbouw computertje

```
; Listing Calclock.asm (calendar and clock)
; lcd-clock plays melody on the hour and chimes hours
; congratulates family and friends on their birthday
;
; artimer register adressen and port b data register
;
arsc1    equ 8d7          ; ar status control register 1
armc     equ 8d5          ; ar mode control register
arrc     equ 8d9          ; ar reload register
arcp     equ 8da          ; ar compare register
drb      equ 8c1          ; port b data register
;
init0    p = clocdis      ; point to the initial clock text
         ld 0,f           ; and load the initial display
         p = initstr      ; point to string with intialize values and..
         v6,va = mp       ; load initialize values into v6...va
         p = drb          ; point to drb and..
         v6,v6 to mp      ; connect servo 1 to pb7 and reset servo 1
         p = armc         ; point to artimer mode control register and..
         v7,v7 to mp      ; set artimer = off and pwm = off
         p= arcp          ; point to artimer compare register and..
         v8,v8 to mp      ; load with f0
;
; v7 = 80,                ; to set artimer = off and pwm = off
; v9 = e0,                ; to set artimer = on and pwm = on
; va = 04,                ; to increment pointer to next music
                          ; table entry
;
mainlo0  skip out 5 = 0 ; skip if chimes flag is not set
         jp chimes        ; jump to chimes
         skip out 6 = 0 ; skip if hours flag is not set
         jp hourwrk       ; jump to hours
mainlo1  skip out f = 1 ; skip if the seconds flag is set
         jp mainlo0
         res out f
         call clock       ; call the clock sub
;
; Test for seconds: 2d...36 (decimal: 45...54), set corresponding
; pointer
;
         v0 = seconds
         p = datsend      ; set pointer to 00 byte
         skip v0 <> 2d    ; from now on, set every second
         p = datstr1      ; the pointer on a dattime string
         skip v0 <> 2e    ; so the tests will be distributed
         p = datstr2      ; in time
         skip v0 <> 2f
         p = datstr3
```

6.7 Chip, een zelfbouw computertje

```
         skip v0 <> 30
         p = datstr4
         skip v0 <> 31
         p = datstr5
         skip v0 <> 32
         p = datstr6
         skip v0 <> 33
         p = datstr7
         skip v0 <> 34
         p = datstr8
         skip v0 <> 35
         p = datstr9
         skip v0 <> 36
         p = datstrA
;
         v0,v0 = mp      ; get first string byte
         skip v0 <> 00
         jp mainlo0      ; if byte = 00, pointer not set jump back
         call dattime    ; compare clock and string
         skip vf <> 00   ; skip if equal
         jp mainlo0      ; not equal, jump
         v0 = 06         ; set pointer to message
         p + v0
         rotate          ; rotate message on display, give beebs
mainlo2  v0 = hours
         skip v0 <> 0a   ; rotate text till 10 am o'clock...
         jp mainlo3
         skip in 0 = 0   ; ...or till button in 0 is pressed
         jp mainlo2
mainlo3  stop rotate
         p = clocdis     ; point to the initial clock text
         ld 0,f          ; and load the initial display
         jp mainlo0
;
hourwrk  vb = rnd, 03    ; vb is random 0...3
         skip vb <> 00   ; set p acording random to melody
         p = melody
         skip vb <> 01
         p = hymne
         skip vb <> 02
         p = stars
         skip vb <> 03
         p = auclair
         call musibox    ; play melody
         v8 = hours      ; get hours into v8
         skip v8 <> 00
         v8 = 0c         ; if hours is null, make hours 12
         vb = 0c
```

6.7 Chip, een zelfbouw computertje

```
         skip v8 > vb   ; double skip to invert skip condition
         skip a
         v8 - vb        ; so this is executed if skip not true
         vb = 30
         vb to timer    ; set timer for delay between melody and chimes
         res out 6      ; reset hours flag, to not get here again
         set out 5      ; set chimes flag to chime hours
         jp mainlo1
;
chimes   skip v8 <> 00
         res out 5      ; reset chimes flag if all hours are chimed
         vb = timer
         skip vb = 00
         jp mainlo1     ; jump back if timer not yet zero
         p = beet       ; point to chime sound
         call musibox   ; chime
         v8 + ff        ; hours = hours - 1
         vb = 20
         vb to timer    ; set timer for delay between chimes
         jp mainlo1
;
clocdis  asciz "   :  :  :  :   "; this is the initital display
initstr  bytes c080f0e004        ; presets for v6...va
;
; Clock, subroutine to show the week, day of week and time on the LCD.
; Note that we use variables to load the display,
; so we can use a loop.
;
clock    v0 = 01        ; initialise v0 and v1
         v1 = 02
         v0 to tone     ; tick every second
clock1   skip v0 <> 01  ; get the corresponding byte
         v3 = weeks     ; and load into v3
         skip v0 <> 04
         v3 = days
         skip v0 <> 07
         v3 = hours
         skip v0 <> 0a
         v3 = minutes
         skip v0 <> 0d
         v3 = seconds
         p = a-stack    ; we let p point to a-stack
         v3 to 3dec     ; and convert v3 to 3 decimals
         p+1            ; we don't need the hundreds
         ld v0,v1       ; show tens and units on display
         v0 + 03        ; let v0 and v1 point
         v1 + 03        ; to next display position
         skip v0 = 10   ; if v0 = 10 we are ready
```

6.7 Chip, een zelfbouw computertje

```
        jp clock1       ; else we loop
        skip v3 = 00
        ret             ; return if v3 <> 00
        v3 = minutes
        skip v3 <> 00
        set out 6       ; set out 6 if minutes = 00 (and seconds = 00)
        ret
;
; dattime, subroutine to compare the running clock against a
; date-time string
; pointed to by the pointer:
; p -> weeks, days, hours, minutes, seconds (all values hex!)
; when a dattim string byte is 7f, it will not be tested
; when the condition is met for the first time, vf will be set as
; flag (<> 00)
; and bit 7 of weeks, date-time string will be set. As soon as
; the condition
; is no more valid this bit wil be reset.
;
dattime v0,v4 = mp      ; move string date and time into v0...v4
        vf = 7f         ; strip bit 7 from v0
        v0 and vf
        skip v0 <> 7f   ; skip if weeks <> 7f
        jp dattim1      ; weeks = 7f, so do not test
        vf = weeks      ; get clock weeks
        skip v0 = vf    ; skip if equal
        jp dattim7      ; not equal, jump
dattim1 skip v1 <> 7f   ; skip if days <> 7f
        jp dattim2      ; days = 7f, so do not test
        vf = days       ; get clock days
        skip v1 = vf    ; skip if equel
        jp dattim7      ; not equal, jump
dattim2 skip v2 <> 7f   ; skip if hours <> 7f
        jp dattim3      ; hours = 7f, so do not test
        vf = hours      ; get clock hours
        skip v2 = vf    ; skip if equal
        jp dattim7      ; not equal, jump
dattim3 skip v3 <> 7f   ; skip if minutes <> 7f
        jp dattim4      ; minutes = 7f, so do not test
        vf = minutes    ; get clock minutes
        skip v3 = vf    ; skip if equal
        jp dattim7      ; not equal, jump
dattim4 skip v4 <> 7f   ; skip if seconds <> 7f
        jp dattim5      ; seconds = 7f, so do not test
        vf = seconds    ; get clock seconds
        skip v4 = vf    ; skip if equal
        jp dattim7      ; not equal, jump
;
```

6.7 Chip, een zelfbouw computertje

```
; dattim string is equal to the clock
; when this happens for the first time, bit 7 of v0 (weeks) will be 0,
; it must be set to 1 and vf must be made <> 00
;
dattim5  v0, v0 = mp
         vf = 80         ; vf is bitmask
         vf and v0       ; strip d6...d0
         skip vf = 00    ; skip if d7 = 0
         jp dattim6      ; jump, this dattim skip has alread been taken
         vf = 80         ; vf is .or. bit 7
         v0 or vf        ; set 7 of v0
         v0, v0 to mp    ; write back to dattim string
         skip a          ; skip always, let vf remain 80
dattim6  vf = 00         ; entry point for skip already taken, res flag
         ret
dattim7  v0, v0 to mp    ; write back to dattim string
         vf = 00         ; reset flag vf
         ret
;
; musibox
; plays music from table, using artimer (pwm-timer)
; before entry pointer must be set on table start
;
musibox save p           ; save pointer, will be needed later
musibo0  rest p          ; point into music table
         vb,ve = mp      ; vb=note, vc=octave, vd=duration, ve=delay
         p + va          ; point to next music table entry
         save p          ; save pointer
         skip vb <> 00   ; skip if note <> 00
         ret             ; ret if note = 00
         p = arrc        ; point to ar reload register..
         vb,vb to mp     ; load note
         p = arsc1       ; point to ar status control register 1 and..
         vc,vc to mp     ; ...load octave into predivider
         vd to timer     ; load duration into timer
         p = armc        ; point to artimer mode control register and..
         v9,v9 to mp     ; set artimer = on and pwm = on
musibo1  vd = timer      ; wait for duration of note
         skip vd = 00
         jp musibo1
         v7,v7 to mp     ; set artimer = off and pwm = off
musibo2  ve + ff         ; wait for delay time between notes
         skip ve = 00
         jp musibo1
         jp musibo0
;
melody   bytes 88a10407
         bytes 81a10407
```

6.7 Chip, een zelfbouw computertje

```
          bytes 88a10407
          bytes 71811007
          bytes 00
;
hymne     bytes 81a10807
          bytes 81a10807
          bytes 88a10807
          bytes 95a10807
          bytes 95a10807
          bytes 88a10807
          bytes 81a10807
          bytes 71a10807
          bytes 60a10807
          bytes 60a10807
          bytes 71a10807
          bytes 81a10807
          bytes 81a11007
          bytes 71a11007
          bytes 81a10807
          bytes 81a10807
          bytes 88a10807
          bytes 95a10807
          bytes 95a10807
          bytes 88a10807
          bytes 81a10807
          bytes 71a10807
          bytes 60a10807
          bytes 60a10807
          bytes 71a10807
          bytes 81a10807
          bytes 71a11007
          bytes 60a11007
          bytes 00
;
stars     bytes 60a11007
          bytes 95a11007
          bytes 88a10407
          bytes 81a10407
          bytes 71a10407
          bytes 60811007
          bytes 95a11007
          bytes 88a10407
          bytes 81a10407
          bytes 71a10407
          bytes 60811007
          bytes 95a11007
          bytes 88a10407
          bytes 81a10407
```

6.7 Chip, een zelfbouw computertje

```
         bytes 88a10407
         bytes 71a11007
         bytes 00
;
auclair  bytes 60a10807
         bytes 60a10807
         bytes 60a10807
         bytes 71a10807
         bytes 81a11007
         bytes 71a11007
         bytes 60a10807
         bytes 81a10807
         bytes 71a10807
         bytes 71a10807
         bytes 60a11007
         bytes 00
;
beet     bytes 88810407
         bytes 88810407
         bytes 71811007
         bytes 00
;
         org 400        ; birthdate strings en congratulations
;
datstr1 bytes 0206      ; weeks = 02d, days = 06d
         bytes 072d     ; hours = 07d, minutes = 45d
         bytes 7f00     ; seconds = don't care, filler
messag1  asciz "Hoera, Willem is jarig!"
;
datstr2 bytes 1102      ; weeks = 17d, days = 02d
         bytes 072d     ; hours = 07d, minutes = 45d
         bytes 7f00     ; seconds = don't care, filler
messag2  asciz "Hoera, Marjan is jarig!"
;
datstr3 bytes 1405      ; weeks = 20d, days = 05d
         bytes 072d     ; hours = 07d, minutes = 45d
         bytes 7f00     ; seconds = don't care, filler
messag3  asciz "Ons Jan Peter is jarig!"
;
datstr4 bytes 1c03      ; weeks = 28d, days = 03d
         bytes 072d     ; hours = 07d, minutes = 45d
         bytes 7f00     ; seconds = don't care, filler
messag4  asciz "Remember onze Reinier! "
;
datstr5 bytes 2105      ; weeks = 33d, days = 05d
         bytes 072d     ; hours = 07d, minutes = 45d
         bytes 7f00     ; seconds = don't care, filler
messag5  asciz "Hoera, Sara is jarig!  "
```

6.7 Chip, een zelfbouw computertje

```
;
datstr6 bytes 2903      ; weeks = 41d, days = 03d
         bytes 072d     ; hours = 07d, minutes = 45d
         bytes 7f00     ; seconds = don't care, filler
messag6  asciz "Hoera, Wil is jarig!   "
;
datstr7 bytes 2a05      ; weeks = 42d, days = 05d
         bytes 072d     ; hours = 07d, minutes = 45d
         bytes 7f00     ; seconds = don't care, filler
messag7  asciz "Hoera, Esther is jarig!"
;
datstr8 bytes 2b04      ; weeks = 43d, days = 04d
         bytes 072d     ; hours = 07d, minutes = 45d
         bytes 7f00     ; seconds = don't care, filler
messag8  asciz "Hoera, Bobsie is jarig!"
;
datstr9 bytes 2e02      ; weeks = 46d, days = 02d
         bytes 072d     ; hours = 07d, minutes = 45d
         bytes 7f00     ; seconds = don't care, filler
messag9  asciz "Tante Andrea is jarig! "
;
datstrA bytes 3203      ; weeks = 50d, days = 03d
         bytes 072d     ; hours = 07d, minutes = 45d
         bytes 7f00     ; seconds = don't care, filler
messagA  asciz "Hoi, Michael is jarig! "
datsend  bytes 00       ; zero byte to check for unchanged pointer
```

Figuur 4/6.7.7-6: De listing van Calclock.asm.

Tot slot

In de secondentests staan heel wat skips, waardoor dit programmadeel wel heel duidelijk is, maar een rechtgeaarde programmeur vraagt zich af of dit niet korter kan. Dat kan vrij eenvoudig met twee optellingen en twee tests van de carry (vf).

De verkregen waarde is de index (0...) in een tabel en moet worden vermenigvuldigd met de offset tussen de entrees van de tabel. Alle felicitatiewensen moeten dan wel even lang zijn, wat met de skipmethode niet het geval is.

Nuttig om te weten is ook dat instructie p + vx een 16 bit optelling is.

(Bob Stuurman)

De listings uit dit hoofdstuk kunt u downloaden van www.vego.nl/chip

4/6.7.8

Chip als "Homesystem"

Inleiding
We gaan onze woning automatiseren met Chip. Belangrijke factoren daarbij zijn de tijd en de temperatuur zowel binnen als buiten, onder andere om de kachel aan of uit te kunnen zetten afhankelijk van temperatuur en tijd. De tijd heeft alles te maken met de klok van Chip en daar kunnen we, dankzij hoofdstuk 4/6.7.7, mee "lezen en schrijven". Ook willen we graag weten of het al donker genoeg is om de lamp aan de doen of dat de lamp juist uit kan. Ons "Chip Homesystem" combineert al deze zaken (en meer) in een juist verband en maakt ons leven iets makkelijker. Bovendien wordt geld bespaard omdat de verwarming niet onnodig aanstaat.

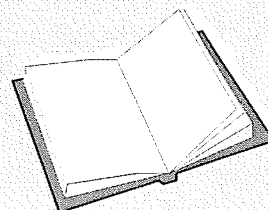
Elektronische temperatuursensor
De temperatuursensor is gebaseerd op de KTY10-6. Dat is een PTC met een nominale weerstand (±1 %) van 2 kΩ bij 25 °C. In de tabel van figuur 4/6.7.8-1 staan de weerstandswaarden voor enkele temperaturen. Let trouwens wel op de 6 in het typenummer want een ander getal duidt op een andere nominale weerstandswaarde.

Het schema
Het schema van de temperatuurmeter is getekend in figuur 4/6.7.8-2. Er zijn twee sensoren noodzakelijk, een voor het buiten-temperatuurbereik van -20 °C tot +40 °C en een voor het binnen-temperatuurbereik van 0 °C tot +40 °C.

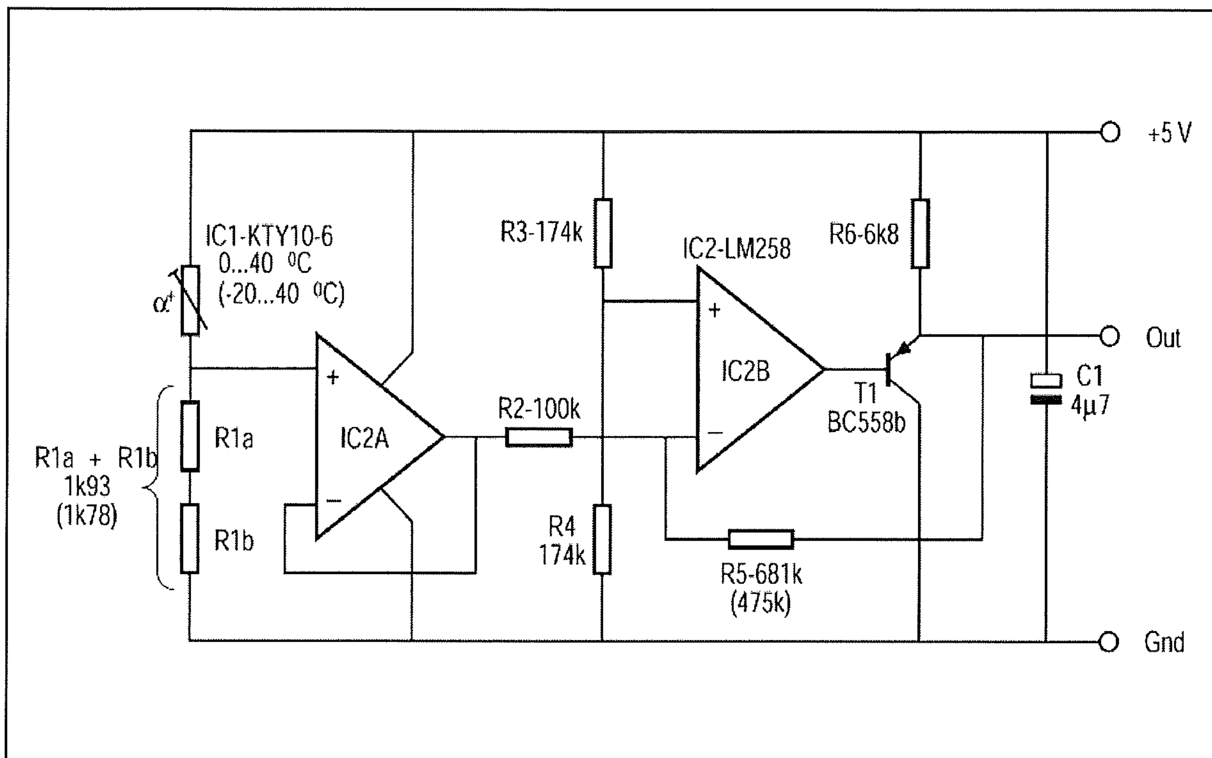
Temp °C	KTY10-6 weerstand in Ω
-20	1.387
-10	1.513
0	1.645
10	1.783
20	1.926
30	2.075
40	2.230

Figuur 4/6.7.8-1: Weerstand van de KTY10-6 sensor bij verschillende temperaturen.

LEES OOK:
Hoofdstuk 4/6.7.1
Hoofdstuk 4/6.7.2
Hoofdstuk 4/6.7.3
Hoofdstuk 4/6.7.7
www.vego.nl/chip



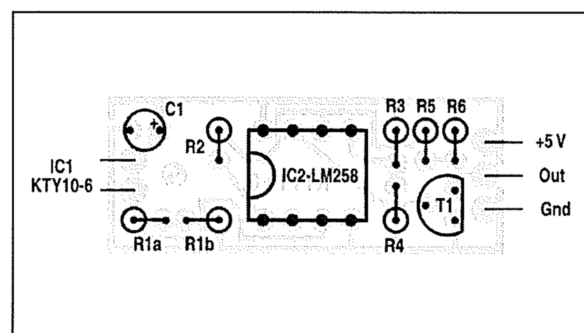
6.7 Chip, een zelfbouw computertje



Figuur 4/6.7.8-2: Het schema van de temperatuursensor.

De sensors zijn identiek op twee weerstandswaarden na, tussen de haakjes zijn die voor de buitensensor opgenomen. In wezen is de sensor in een brugschakeling opgenomen, waarbij het spanningsverschil in de brug wordt versterkt. Het IC (LM258) heeft als uitgangstrap een NPN-emittervolger. Om het spanningsbereik van de uitgang groter te maken, is T1 opgenomen.
De "middentemperatuur" voor de binnensensor is +20 °C en voor de buitensensor +10 °C. De brug is dan in evenwicht (zie ook de tabel van figuur 4/6.7.8-1) en de uitgangsspanning is de halve voedingsspanning. De sensor is bij Chip vrij ongevoelig voor schommelingen in de voedingsspanning en, omdat slechts een deel van de temperatuur/weerstand-curve van de KTY10-6 wordt gebruikt, vrijwel lineair.

De bouw van de schakeling
Figuur 4/6.7.8-3, op de laatste pagina van dit hoofdstuk, laat de print zien en figuur 4/6.7.8-4 de componentenopstelling. Gebruik bij voorkeur 1 % metaalfilm weerstanden, die zijn het minst temperatuurgevoelig. Om de sensor zo klein mogelijk te maken zijn alle weerstanden rechtop gemonteerd.



Figuur 4/6.7.8-4: De componentenopstelling van de temperatuursensor.

6.7 Chip, een zelfbouw computertje

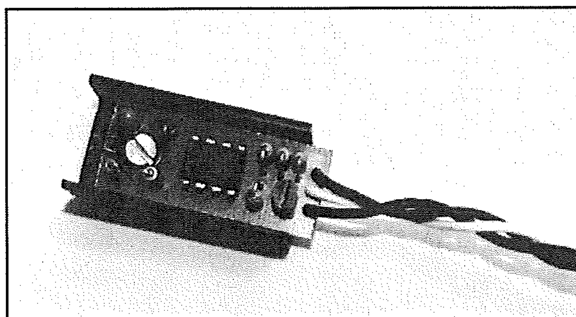
ONDERDELENLIJST

WEERSTANDEN, 1/4 W, 5 %

R1		1,93 kΩ
R1'		1,78 kΩ
R2		100 kΩ
R3,R4		174 kΩ
R5		681 kΩ
R5'		475 kΩ
R6		6,8 kΩ
CONDENSATOREN		
C1	4,7 µF	16 V printelco
HALFGELEIDERS		
T1		BC558B
IC1		KTY10-6
IC2		LM258

De montage van de print

Het printje past in het U-vormig koelelement voor TO-220 halfgeleiders, zie figuur 4/6.7.8-5. De sensor wordt aan zijn draden omgebogen zodat hij vlak op de soldeerzijde ligt. Aan de aansluitkant komt als afstandstuk tussen de print en het koelelement een strookje epoxy en dan wordt het printje met een M3 boutje in het koelelement vastgezet, waarbij de sensor wordt vastgeklemd tussen de print en de koelvin. De lengte van het sensorsnoer is niet kritisch, bij onze opstelling was dat voor de buitensensor circa 5 meter.



Figuur 4/6.7.8-5: De montage van de sensor en de print.

Opmerking

Het is mogelijk voor de binnensensor een LM358 (0 °C tot 70 °C) te gebruiken. Dat kan ook voor de buitensensor, als men het printje binnenskamers houdt en de sensor via een snoertje verbindt. De LM258 heeft namelijk een groter temperatuurbereik (-25 °C tot +85 °C).

Meten van de binnentemperatuur

De listing van figuur 4/6.7.8-6 toont het programma Tempin.asm om de, door de binnensensor op input 1, gemeten waarde om te rekenen naar °C en op het display te zetten. Heel aardig is dat kan worden afgetrokken door op te tellen. Het schalen van de waarde is heel eenvoudig door Chip's 16 bit vermenigvuldig- en deelinstructies.

Meten van de buitentemperatuur

Het programma Tempout.asm voor de buitensensor, voorgesteld in figuur 4/6.7.8-7, is iets ingewikkelder omdat nu natuurlijk ook een negatieve temperatuur mogelijk is.

6.7 Chip, een zelfbouw computertje

```
; Listing Tempin.asm
;
main      skip out f = 0
          call tempin
          jp main
;
; tempin, subroutine to measure interior temperature with kty 10-6
; chip temp-in sensor connected to input 1
;
tempin    res out f        ; reset seconds flag
          p = tempin1      ; point to text
          ld 0,f           ; load display
          v0 = ana 1       ; get raw temperature
          v0 + c6          ; subtract 3a (temp @ 0 øC) by adding c6
          p = a-stack      ; point to a-stack
          v1 = 18          ; scaling factor is 18h/53h
          v0 * v1 to mp    ; multiply by 18...
          v1 = 53
          v0 = mp/v1       ; and divide by 53
          v0 to 3dec mp    ; convert to decimals
          p + 1            ; point to tens
          ld b,c           ; load display
          ret
tempin1   asciz "Temp in  =    oC"
; note: change 6F (o) in hexfile into LCD degree sign
```

Figuur 4/6.7.8-6: De listing van Tempin.asm.

Als dat het geval is, wordt een minteken op het display gezet en wordt de absolute waarde van de temperatuur genomen.

Nauwkeurigheid

Door de initiële nauwkeurigheid van de KTY10-6 en de 1 % metaalfilmweerstanden in de sensorschakeling zal de temperatuur vrij goed kloppen. Als dat niet het geval is, is het "ijken" van de sensor een interessant klusje voor een regenachtige zondag.

De hardware van het "Chip Homesystem"

In figuur 4/6.7.8-8 is de hardware van ons "Chip Homesystem" voorgesteld. Op output 0 is via een schakeltransistor een relais aangesloten voor sturing van een schemerlamp. Op output 1 een identieke schakeling voor sturing van een alarmlamp en een luide buzzer. Tenslotte is servo 1 beschikbaar om de kachel uit of aan te zetten. Het is niet mogelijk om voor de kachelbediening een standaard recept te geven omdat dat per geval kan verschillen. Bij ons bleek de draaiknop van de radiatorkraan een pal in te drukken of los te laten. Op de kraan hebben we een hefboomconstructie gemonteerd waarbij de servo via de hefboom de pal indrukt of loslaat. Belangrijk is wel, dat de pal door de servo helemaal wordt ingedrukt of losgelaten.

6.7 Chip, een zelfbouw computertje

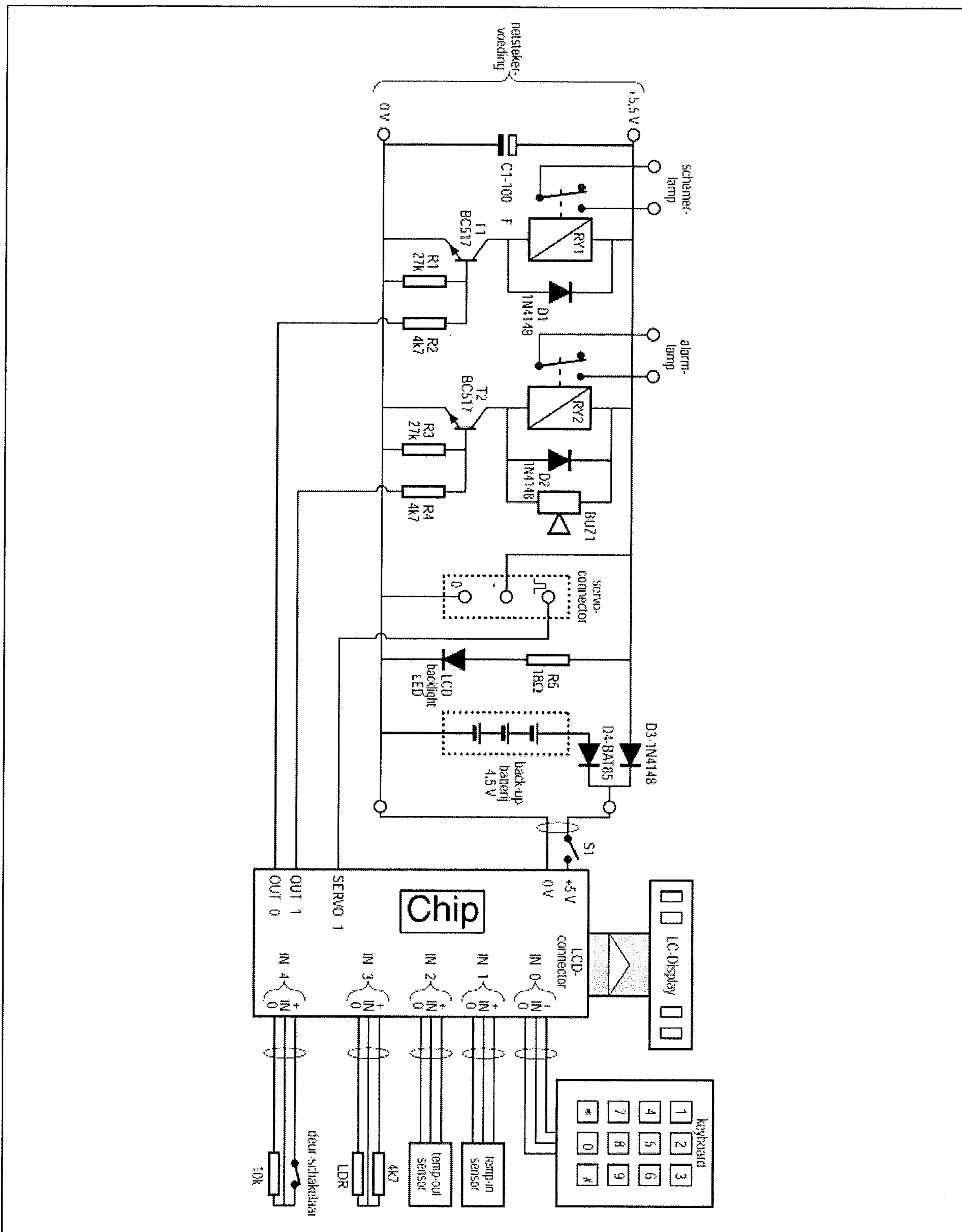
```
; Listing Tempout.asm
;
main     skip out f = 0
         call tempout
         jp main

; tempout, subroutine to measure exterior temperature with kty 10-6
; chip temp-out sensor connected to input 2
;
tempout  res out f       ; reset seconds flag
         p = tempou2     ; point to text
         ld 0,f          ; load display
         v0 = ana 2      ; get raw temperature
         v0 + 98         ; subtract 68h (temp 0 øC) by adding 98h
         skip vf = 00    ; skip if below 0 øC
         jp tempou1
         v1 = 00         ; v0 = abs(v0)
         v1 - v0         ; note: 00h - e0h = 20h
         v1 to v0
         p = tempou3     ; point to minus sign and ...
         ld a,a          ; load display
tempou1  p = a-stack     ; point to a-stack
         v1 = 19         ; scaling factor is 19h/3fh
         v0 * v1 to mp   ; multiply by 19...
         v1 = 3f
         v0 = mp/v1      ; and divide by 3f
         v0 to 3dec mp   ; convert to decimals
         p + 1           ; point to tens
         ld b,c          ; load display
         ret
tempou2  asciz "Temp out=     oC"
tempou3  asciz "-"
; note: change 6F (o) in hexfile into DF (LCD degree sign)
```

Figuur 4/6.7.8-7: De listing van Tempout.asm.

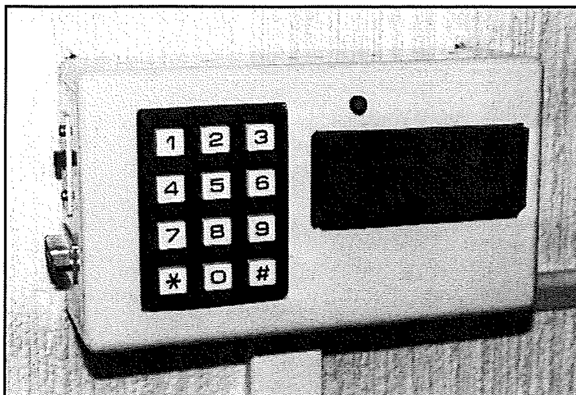
De servo mag daarbij niet mechanisch tegen zijn eindblokkeringen oplopen. Dan zou de servomotor stroom blijven trekken en niet lang meegaan en ook kan de voeding worden overbelast.
Alles, inclusief de backlight LED van het LCD, wordt gevoed door een gestabiliseerde netstekervoeding die op 5,5 V is afgeregeld. De Chip homecomputer wordt normaal door de netstekervoeding gevoed via diode D3, maar bij uitval van de netspanning door de 4,5 V batterij via D4. Het backlight wordt via een serieweerstand van 10 Ω, 1 W direct door de netstekervoeding gevoed. Om aardlussen te voorkomen zijn alleen de actieve pennen van out 0, out 1 en servo 1 aangesloten. De 0 V loopt via de voeding.

6.7 Chip, een zelfbouw computertje



Figuur 4/6.7.8-8: De volledige elektronica van het "Chip Homesystem".

6.7 Chip, een zelfbouw computertje



Figuur 4/6.7.8-9: De behuizing van het "Chip Homesystem".

Op input 0 is het keyboard aangesloten, op input 1 de sensor voor de binnentemperatuur, op input 2 die voor de buitentemperatuur, op input 3 een LDR die het buitenlicht meet en op input 4 een buitendeur schakelaar, zo'n type met een reedswitch, die door een magneet op de deur gesloten wordt gehouden en open gaat als de deur wordt geopend. In dat geval wordt input 4 naar laag getrokken door de 10 kΩ weerstand.

De behuizing van het systeem

Omdat wij ons Chip experimenteersysteem nodig hebben voor het realiseren van andere toepassingen, hebben we nog een Chip gebouwd en die samen met een LCD (met backlight LED) en een keyboardje netjes in een OKW standaardbox 1 met een lengte van 150 mm gemonteerd, zie figuur 4/6.7.8-9. De Chip-print zit met twee zeskant afstandstukken op de bodem van het kastje vast, met tussen de bodem en de afstandstukken een aluminiumplaatje, dat door de metalen afstandstukken verbonden is met de "massa" van de Chip-print. In de linker zijkant van het kastje zijn een aan/uit-schakelaar en een connector voor de seriële verbinding opgenomen.

De homecomputer is op een goed bereikbare plaats op ooghoogte tegen de wand bevestigd.
Recht eronder, iets boven de plint, is eenzelfde kastje voor de opname van aanvullende elektronica gemonteerd. Tussen de kastjes zit een kabelgoot en in de kastjes zijn uitsparingen gemaakt voor de bedrading via de kabelgoot.

Homesys software

Homesys.asm, zie de listing van figuur 4/6.7.8-10, is in wezen een combinatie van listings die besproken zijn. Voor de aansturing van het display wordt een aparte teller gebruikt, die door output f eenmaal per seconde wordt verhoogd. Hierdoor is een "display verdeling" in de tijd verkregen. Eerst wordt gedurende drie seconden de actuele tijd getoond, dan gedurende een seconde elk, de binnentemperatuur, gevolgd door de buitentemperatuur en de gemeten waarde van het daglicht.
Als de kachel aanstaat wordt de display cyclus besloten met "Heater is active". Als basis voor het regelen van de kachel wordt een aparte variabele gebruikt temp set. De temperatuur die door de binnensensor is gemeten, wordt vergeleken met temp set. Als de temperatuur lager is wordt de kachel aangezet, is hij hoger, dan wordt hij uitgezet. Temp set wordt door drie tijdstrings geregeerd. 's Morgens vroeg wordt hij op 25 °C ingesteld, om 8 uur op 20 °C en s avonds weer wat hoger.
Afhankelijk van de hoeveelheid buitenlicht én de tijd wordt de schemerlamp geregeld. 's Nachts is hij altijd uit, overdag wordt hij ingeschakeld als het buiten donker is. Dus als overdag zwarte wolken de lucht gaan bedekken, gaat de lamp aan.

6.7 Chip, een zelfbouw computertje

```
; Listing Homesys.asm
; program to automate the house-heating, house-lighting,
; intruder-alarm
; includes running clock with weeks, days, hours, minutes and seconds
; also an external temperature sensor and an eggtimer.
; new functions can easily be added.
;
; use of inputs, outputs and variables
;
; in 0   = keyboard
; in 1   = temp in sensor
; in 2   = temp out sensor
; in 3   = LDR sensor
; in 4   = door switch (alarm)
; out 0  = relay 1 = light
; out 1  = relay 2 = alarm
; out 4  = continuity check
; out 6  = rotate text is active
; out 7  = waiting time flag before setting alarm
; out 8  = alarm on/off
; out 9  = light automatic on/off
; out a  = heater on flag
; out b  = eggtimer on flag
; out c  = wait-on flag
; out d  = key down flag
; out e  = select flag
; out f is set every second by the operating system
; v0-v4  = general purpose variables
; v7     = light value low
; v8     = light value high
; v9     = temp low
; va     = temp medium
; vb     = temp high
; vc     = temp set
; vd     = time distributor
; ve     = selector
; vf     is carry/borrow/overflow flag
;
weekvar  equ 8aa         ; internal weeks register
;
start    set out 9       ; set light automatic on
         p = initvar
         v7,ve = mp      ; initialise v7 - ve
         son             ; start servo drive
main_0   set out 4       ; set out 4 for checking with scoop
         res out 4       ; reset out 4 for checking with scoop
         skip out 8 = 1  ; skip if alarm flag is on
         jp main_1
         skip in 4 = 0   ; skip if house entry door switch activated
         jp main_1
         set out 1       ; activate alarm
         v0 = 02
```

6.7 Chip, een zelfbouw computertje

```
         v0 to mintimer ; load minutes timer
main_1   skip out d = 0 ; skip if .not. key down
         jp keydown     ; key beeing pressed, jump to keydown
         skip out e = 0 ; skip if .not. select active
         jp select1     ; select active, jump to select
         skip ve <> ff  ; skip if ve .not. active (= ff)
         jp main_3      ; select .not. active, jump to continue
                        ; main loop
         jp getkeyn     ; select is active, try to get a key value
main_2   skip ve <> 00  ; jump according to selector value ve
         jp settem1
         skip ve <> 10
         jp setegg1
         skip ve <> 40
         jp settim1
main_3   v0 = key 0     ; get a key value
         skip v0 <> 3a  ; skip if .not. 3a (= *)
         jp select0     ; got a key value 3a, jump to select0
         skip out f = 1 ; skip if on the seconds flag
         jp main_0      ; no seconds flag, loop to main_0
;
; distribution in time (every second)
;
         res out f      ; reset seconds flag
         call checkDT   ; check date-time strings and if necessary
                        ; take action
         skip out 6 = 0 ; skip if rotating text is off
         jp rotatin
secs_0   skip out b = 1 ; skip if eggtimer is active
         jp secs_1      ; jump to continue distribution in time
         call eggshow   ; show eggtimer value
         jp main_0      ; and loop to main
secs_1   vd + 01        ; increment time distributor vd
         skip vd = 01   ; perform the task as set by vd
         jp secs_2
         p = clock2     ; point to the initial clock text
         ld 0,f         ; and load the initial clock display
         call clock     ; call the clock sub
         jp main_0
secs_2   skip vd = 02
         jp secs_3
         call clockSM   ; refresh clock seconds and minutes
         skip out 1 = 1 ; skip if alarm has been activated
         jp main_0
         v0 = mintimer  ; get minutes timer
         skip v0 <> 00  ; skip if not yet zero
         res out 1      ; zero, set alarm off
         jp main_0
secs_3   skip vd = 03
         jp secs_4
         call clockSM   ; refresh clock seconds and minutes
         skip out 7 = 1 ; skip if alarm activate waiting time flag
```

6.7 Chip, een zelfbouw computertje

```
          jp main_0
          v0 = mintimer   ; get minutes timer
          skip v0 = 00    ; skip if zero
          jp main_0       ; not yet zero
          res out 7       ; reset alarm activate waiting time flag ...
          set out 8       ; ... and set alarm to sharp
          jp main_0
secs_4    skip vd = 04
          jp secs_5
          call clockSM    ; refresh clock seconds and minutes
          call birtday    ; check for birthday, if so, congratulate
          jp main_0
secs_5    skip vd = 05
          jp secs_6
          call tempin     ; call show interior temperature
          jp main_0
secs_6    skip vd = 06
          jp secs_7
          call tempout    ; call show exterior temperature
          jp main_0
secs_7    skip vd = 07
          jp secs_8
          call daylite    ; call show indicative external light value
          skip out a = 1  ; skip if heater on flag
          vd = 00         ; show clock again if heater is off,
          jp main_0
secs_8    p = heatonT     ; else show heaton text
          ld 0,f
          vd = 00
          jp main_0
heatonT   asciz "Heater is active"
;
rotatin   v0 = mintimer   ; get minutes timer
          skip v0 = 00    ; skip if run out
          jp main_0       ; still running, jump to main
          res out 6       ; reset rotate active flag
          stop rotate     ; disable rotate and reset display
          vd = 00         ; set seconds sequencer to phase 0
          jp main_0       ; jump to normal sequence display
;
; keydown, jump routine to wait for key release or time out
;
keydown   v0 = key 0      ; v0 is key value
          skip v0 = 3f    ; skip if no key pressed
          jp keydwn1      ; jump, key still beeing pressed
          res out d       ; key has been released, reset key down
          jp main_0       ; and jump to main_0
keydwn1   v0 = timer      ; get time out value
          skip v0 = 00
          jp main_0       ; time still running, jump to main_0
          res out c       ; reset wait on flag
          res out e       ; reset select flag
```

6.7 Chip, een zelfbouw computertje

```
         vd = 00         ; reset time distributor
         ve = ff         ; set selector ve to off
         v0 = 02         ; give a sound signal
         v0 to tone      ; while key beeing pressed
         jp main_0       ; and loop to main_0
;
; getkeyn, jump routine to get a numerical key value for passing to
; selected
; jump routine main_2 according to selector ve
;
getkeyn  v0 = key 0      ; get a key value into v0
         v1 = 39
         skip v0 > v1    ; skip if .not. numerical (0-9)
         jp getkey1      ; jump, got numerical value
         v0 = timer      ; get time out value
         skip v0 = 00    ; skip on time out
         jp main_0       ; not yet time out, jump to try to get
                         ; good value
         ve = ff         ; set selector ve to off
         vd = 00         ; set time distributor to show clock
                         ; (load mask)
         jp main_0       ; and continue main task
getkey1  set out d       ; got key value, set key down
         v1 = 03
         v1 to tone      ; give key beep
         v1 = 8c         ; circa 5 seconds
         v1 to timer     ; set time out value
         jp main_2       ; jump to selected set routine in main_2
;
; select, jump routine to make a choice out of rotating menus
;
select0  v0 = 02
         v0 to tone      ; give a key beep
         vd = 00         ; reset time distributor
;        vd to mintimer  ; set minutes timer to zero and ...
         res out 6       ; reset rotate text flag in case ...
         stop rotate     ; text was rotating
         res out b       ; disable eggtimer
         p = seltxt?     ; display the ?
         ld f,f          ; at position f
         p = seltxt0     ; set p to the first of the selections
         ld 0,e          ; load chars 0-e
         ve = 00         ; preset the selector to 00
         set out e       ; set selection to active
         set out d       ; set key down to active
         v0 = 3c
         v0 to timer     ; load time out value
         jp main_0       ; jump back to main_0
select1  skip out c = 0  ; entry point for rotate selections,
         jp select2      ; jump if time out value already loaded
         v0 = 3c         ; c = 0, load time out value
         v0 to timer
```

6.7 Chip, een zelfbouw computertje

```
        set out c       ; and set c
select2 v0 = key 0      ; get a key value
        skip v0 <> 3a
        jp select3      ; jump to select3 if key value = 3a (*)
        skip v0 <> 3b
        jp select4      ; jump to select4 if key value = 3b (#)
        v0 = timer      ; get time out value
        skip v0 = 00    ; skip if time is out
        jp main_0       ; not yet, keep trying to get 3a or 3b
        res out e       ; time is out, reset selector (no more active)
        res out c       ; reset wait on
        jp main_0       ; and loop to perform main tasks
select3 v0 = 02         ; got key value 3a (*)
        v0 to tone      ; give key beep
        skip ve <> 40   ; skip if .not. last selector value
        ve = f0         ; yes, last, preset to f0
        ve + 10         ; add 10 to selector value (last will become
00)
        p = seltxt0     ; set pointer to first slection text
        p + ve          ; add selector offset to p
        ld 0,e          ; and put text on display
        res out c       ; reset wait on flag
        set out d       ; set keydown flag
        v0 = 3c
        v0 to timer     ; load time out value
        jp main_0       ; loop to cycle through selection while .not.
                        ; time out
select4 v0 = 02         ; got a key value 3b (#)
        v0 to tone      ; give key beep
        res out c       ; reset wait on
        res out e       ; reset selector flag (disable jump to select)
        set out d       ; set key down flag
        v0 = 6c
        v0 to timer     ; set time-out value
        skip ve <> 00   ; and jump to entry point of set routines
        jp settemp      ; according to selector value ve
        skip ve <> 10
        jp seteggt
        skip ve <> 20
        jp setlite
        skip ve <> 30
        jp setalar
        skip ve <> 40
        jp settime
        break           ; never reach this point
seltxt0 asciz "Set temperature"; selector ve = 00
        asciz "Start egg-timer"; selector ve = 10
        asciz "Set light on/of"; selector ve = 20
        asciz "Set alarm on/of"; selector ve = 30
        asciz "Set date & time"; selector ve = 40
seltxt? asciz "?"
;
```

6.7 Chip, een zelfbouw computertje

```
; setalar, jump routine to set alarm on or off
;
setalar  skip out 8 = 1 ; skip if alarm on
         skip out 7 = 0 ; skip if waiting time flag
         jp setala1     ; jump to set alarm off
         set out 7      ; set alarm waiting time flag on
         v0 = 05
         v0 to mintimer ; set waiting time before setting alarm to
                        ; 5 minutes
         p = setala4    ; point to text on
         jp setala2
setala1  res out 7      ; set waiting time flag off
         res out 8      ; set alarm flag to off
         res out 1      ; set alarm off
         v0 = 00
         v0 to mintimer ; clear minutes timer
         v0 to sectimer
         p = setala3    ; point to text off
setala2  ld 0,f         ; put on display
         ve = ff        ; set selector ve to off
         vd = 00        ; reset time distributor (to load clock mask
                        ; and clock)
         jp main_0      ; jump to main task
setala3  asciz "alarm = set off!"
setala4 asciz "alarm = set on! "
;
; setlite, jump routine to set light on or off
;
setlite  skip out 0 = 0 ; skip if light off
         jp setlit1     ; jump to set light off
         res out 9      ; set light automatic to off
         set out 0      ; set light on
         p = setlit4    ; point to text on
         jp setlit2
setlit1  res out 9      ; set light automatic to off
         res out 0      ; set light off
         p = setlit3    ; point to text off
setlit2  ld 0,f         ; put on display
         ve = ff        ; set selector ve to off
         vd = 00        ; reset time distributor (to load clock mask)
         jp main_0      ; jump to main task
setlit3  asciz "light = set off!"
setlit4 asciz "light = set on! "
;
; seteggt, jump routine to load the eggtimer
;
seteggt  p = setegg4    ; entry point from select, point to mask
         ld 0,f         ; display mask
         p = a-stack    ; point p to a-stack
         v0 = 30        ; and load hundreds with 30 = ascii 0
         v0,v0 to mp
         p + 1          ; set p to tens
```

6.7 Chip, een zelfbouw computertje

```
        v2 = 09          ; set v2 as index to tens of minutes
        jp main_0        ; jump to main to get a key value (or time out)
setegg1 v0,v0 to mp      ; load key value into mp + 1 (tens)
        ld v2,v2         ; put tens on display according index v2
        p + 1            ; increment pointer
        v2 + 01          ; increment index v2
        skip v2 <> 0b
        jp setegg2       ; jump if index v2 = 0b
        skip v2 <> 0f
        jp setegg3       ; jump if index v2 = 0f
        jp main_0        ; not 0b, not 0f, jump to main to get next key
value
setegg2 p = a-stack      ; index = 0b,
        v3 = 3dec mp     ; save minutes in v3
        p + 1            ; point p to tens
        v2 = 0d          ; adjust index v2 to tens of seconds on display
        jp main_0        ; jump to main to get next key values
setegg3 p = a-stack      ; index = 0f
        v0 = 3dec mp     ; convert key values to byte into v0
        v0 to sectimer   ; and load into seconds timer
        v3 to mintimer   ; load minutes timer
        ve = ff          ; set selector ve to off
        set out b        ; set out b as flag for eggtimer loaded
        jp main_0        ; resume main task
setegg4 asciz "Eggtime: ??m ??s"
;
; settemp, jump routine to display/set the 'set' temperature
;
settemp p = settem2      ; entry point from select, point to mask
        ld 0,f           ; display mask
        p = a-stack      ; convert vc (= set temperature)
        vc to 3 dec mp   ; to decimal on the a-stack
        p + 1            ; point to tens (hundreds will be
                         ; 30h = ascii 0)
        ld b,c           ; display tens and units
        v2 = 0b          ; set v2 as index to display 0b
                         ; (tens of minutes)
        jp main_0        ; jump to main to get key value (or time out)
settem1 v0,v0 to mp      ; load key value into mp + 1 (tens)
        ld v2,v2         ; load on display via index v2
        p + 1            ; point to units
        v2 + 01          ; point index v2 to units
        skip v2 = 0d     ; skip if we got tens and units
        jp main_0        ; not yet ready, jump back to main
        p = a-stack      ; got two values
        vc = 3dec mp     ; convert to byte and load into vc
                         ; (new 'set' temp)
        ve = ff          ; set selector ve to off
        vd = 00          ; reset time distributor (to load clock mask)
        jp main_0        ; jump to main task
settem2 asciz "Temp set =   ²C"
; note: change 6F (o) in hexfile into DF (LCD degree sign)
```

6.7 Chip, een zelfbouw computertje

```
;
; settime, jump routine to display/set the date & time
;
settime  p = settimT     ; point to date/time mask
         ld 0,f          ; put on display
         p = a-stack     ; point p to a-stack
         v0 = 30         ; and load hundreds with 30 = ascii 0
         v0,v0 to mp
         p + 1           ; set p to tens
         v2 = 00         ; nibble counter = 00
         v3 = 00         ; display index counter = 00
         v4 = 00         ; byte index counter = 00
         jp main_0
settim1  v0,v0 to mp     ; load key value into mp + 1 (tens)
         v2 + 01         ; increment nibble counter
         v3 + 01         ; increment display index counter
         ld v3,v3        ; load on display via index v2
         p + 1           ; point to units
         v0 = 01         ; is the nibble counter odd
         v0 and v2       ; get only bit 0
         skip v0 = 00    ; skip if even
         jp main_0       ; yes, odd get low nibble
         p = a-stack
         v0 = 3dec mp    ; convert to byte
         p = settimS     ; point to start of temporarily storage place
         p + v4          ; point to storage position
         v0,v0 to mp     ; store byte
         v4 + 01         ; increment byte index counter
         v3 + 01         ; increment display index counter
         p = a-stack
         p + 1           ; point to tens on a-stack
         skip v4 = 05    ; skip if we got 5 bytes
         jp main_0
         p = settimS     ; point to start of storage
         v0,v4 = mp      ; copy bytes into v0-v4
         p = weekvar     ; point to clock weeks internal register
         v0,v4 to mp     ; copy variables into internal regs
         ve = ff         ; set selector ve to off
         vd = 00         ; reset time distributor (to load clock mask)
         jp main_0       ; jump to main task
settimT  asciz " ??:??:??:??:?? "; date & time mask
settimS  bytes 0011223344; temporarily storage place for bytes
;
; subroutine checkDT, compares date-time strings against
; the running clock
; and takes the necessary action when equal.
;
checkDT  p = DTliteS     ; point to set light automatic on DT-string
         call dattime    ; compare
         skip vf = 00    ; skip not equal
         set out 9       ; set light automatic flag on
         p = DTliteR     ; point to set light automatic off DT-string
```

6.7 Chip, een zelfbouw computertje

```
         call dattime    ; compare
         skip vf <> 00   ; skip if equal
         jp checkD1      ; jump next
         res out 9       ; set light automatic flag off
         res out 0       ; set light off
checkD1  p = DTtempH     ; point to set temperature high DT-string
         call dattime    ; compare
         skip vf = 00    ; skip not equal
         vb to vc        ; copy high temperature into temp set
         p = DTtempM     ; point to set temperature medium DT-string
         call dattime    ; compare
         skip vf = 00    ; skip not equal
         va to vc        ; copy medium temperature into temp set
         p = DTtempL     ; point to set temperature low DT-string
         call dattime    ; compare
         skip vf = 00    ; skip not equal
         v9 to vc        ; copy low temperature into temp set
         ret
;
; Date Time string to set light automatic on, every day at 06.00
;
DTliteS  bytes 7f7f      ; weeks, days; note: 7f = does't care
         bytes 0600      ; hours, minutes
         bytes 7f        ; seconds
;
; Date Time string to set light automatic off, every day at 19.00
;
DTliteR  bytes 7f7f      ; weeks, days
         bytes 1300      ; hours, minutes
         bytes 7f        ; seconds
;
; Date Time string to set temperature to high, every day at 05.30
;
DTtempH  bytes 7f7f      ; weeks, days
         bytes 051e      ; hours, minutes
         bytes 7f        ; seconds
;
; Date Time string to set temperature to medium, every day at 08.00
;
DTtempM  bytes 7f7f      ; weeks, days
         bytes 0800      ; hours, minutes
         bytes 7f        ; seconds
;
; Date Time string to set temperature to low, every day at 18.30
;
DTtempL  bytes 7f7f      ; weeks, days
         bytes 121e      ; hours, minutes
         bytes 7f        ; seconds
;
; birtday, subroutine to show birthday rotating text on display
; because weeks and days are used these must be adjusted every year
;
```

6.7 Chip, een zelfbouw computertje

```
birtday  p = DTbirt1     ; point to birthday date-time 1
         call dattime    ; compare
         skip vf <> 00   ; skip if equal
         ret             ; else return
         set out 6       ; set rotating flag
         p = birttx1     ; point to birthday text
         rotate          ; start rotating
         v0 = 05
         v0 to mintimer  ; for 5 minutes
         ret
;
birttx1  asciz "Congratulations Bob!"
;
DTbirt1  bytes 2b02      ; weeks, days; Week 43, day 2, 08:10:00
         bytes 080a      ; hours, minutes
         bytes 7f        ; seconds
;
; eggshow, subroutine to show minutes and seconds timer
; the eggtimer mask has already been displayed by seteggt
;
eggshow  v0 = mintimer   ; get minutes into v0
         p = a-stack
         v0 to 3dec mp   ; convert to decimal on the a-stack
         p + 1           ; point to tens
         ld 9,a          ; display tens and units
         v1 = sectimer   ; get seconds into v1
         p = a-stack
         v1 to 3dec mp   ; convert to decimal on the a-stack
         p + 1           ; point to tens
         ld d,e          ; display tens and units
         v0 or v1        ; v0 = v0 .or. v1
         skip v0 = 00    ; skip if both zero
         ret             ; not yet zero, return
         res out b       ; reset eggtimer active flag
         vd = 00         ; set time distributor to 00 (load clock mask)
         v0 = ff         ; alert user that
         v0 to tone      ; time has run out
         v0 = 03         ; because the beeper is not always heard
         v1 = 01         ; we use the alarm to notify the user
         v0 to sectimer
         v1 to mintimer
         set out 1
         ret
;
; clock, subroutine to show the week, day of week and time on the LCD.
; Note that we use variables to load the display,
; so that we can use a loop.
;
clock    v0 = 01         ; initialise v0 and v1
         v1 = 02
clock1   skip v0 <> 01   ; get the corresponding byte
         v3 = weeks      ; and load into v3
```

6.7 Chip, een zelfbouw computertje

```
         skip v0 <> 04
         v3 = days
         skip v0 <> 07
         v3 = hours
         skip v0 <> 0a
         v3 = minutes
         skip v0 <> 0d
         v3 = seconds
         p = a-stack      ; we let p point to a-stack
         v3 to 3dec       ; and convert v3 to 3 decimals
         p + 1            ; we don't need the hundreds
         ld v0,v1         ; show tens and units on display
         v0 + 03          ; let v0 and v1 point
         v1 + 03          ; to next display position
         skip v0 = 10     ; if v0 = 10 we are ready
         jp clock1        ; else we loop
         ret
clock2   asciz "   :  :  :  :    "; this is the initital display
;
clockSM  v0 = seconds     ; this short routine refreshes only
         p = a-stack      ; the seconds and minutes, in order
         v0 to 3dec mp    ; to save time
         p + 1            ; probably nobody will notice the hours...
         ld d,e
         v0 = minutes
         p = a-stack
         v0 to 3dec mp
         p + 1
         ld a,b
         ret
;
; tempin, subroutine to measure interior temperature with kty 10-6.
; chip sensor connected to input 1.
; Midpoint is 20 øC, so the range is 0-40 øC. At 20 øC Rkty
; 10-6 = 1926 ê
; The formulae for the temperature in øC: (raw - 3dh) * 18h/53h
; (see curve).
;
tempin   p = temitxt      ; set pointer to temperature in mask
         ld 0,f           ; display mask
         v0 = ana 1       ; get raw temperature
         v0 + c3          ; subtract 3dh (temp @ 0 øC) by adding c3h
         p = a-stack      ; set p to a-stack
         v1 = 18          ; and
         v0 * v1 to mp    ; multiply raw by 18h
         v1 = 53          ; then
         v0 = mp/v1       ; divide by 53h
         v0 to 3dec mp    ; convert to decimal
         p + 1            ; set p to tens
         ld b,c           ; display temp in øC
;
; check measured temperature (ist_value) against set
```

6.7 Chip, een zelfbouw computertje

```
; temperature (sol_value),
; when ist_val < sol_val then set heater on,
; when ist_val > sol_val then set heater off
; when ist_val = sol_val then do nothing (hysteresis)
;
        skip v0 <> vc   ; skip ist_val <> sol_val
        ret             ; equal, return nothing to do
        v0 - vc         ; ist_val = ist_val - sol_val
        skip vf = 00    ; skip if ist_val > sol_val (vf = borrow)
        jp tempin1      ; ist_val < sol_val, jump to set heater on
        call resheat    ; call sub to set heater off
        ret
tempin1 call setheat    ; call sub to set heater on
        ret
temitxt asciz "Temp in =     ²C"
; note: change 6F (o) in hexfile into DF (LCD degree sign)
;
; setheat and resheat, subroutines to set heater on if not already on,
; or off if not already off, the heater motor is servo 1
; out a  = heater on flag
;
setheat skip out a = 0 ; skip if heater is off
        ret             ; already on, return
        set out a       ; set heater on flag
        v0 = ff
        v0 to s1        ; and rotate heater servo
        ret
;
resheat skip out a = 1 ; skip if heater is on
        ret             ; already off, return
        res out a       ; reset heater on flag
        v0 = 0a
        v0 to s1        ; and rotate heater servo
        ret
;
; tempout, subroutine to measure exteriour temperature with kty 10-6
; chip sensor connected to input 2
; Midpoint is 10 øC, so the range is -20 - +40 øC. At 10 øC
; Rkty 10-6 = 1783 ê
; The formulae for the temperature in øC: (raw - 68h) * 19h/3fh
; (see curve).
;
tempout p = temotxt     ; set pointer to temperature out mask
        ld 0,f          ; display mask
        v0 = ana 2      ; get raw temperature
        v0 + 98         ; subtract 68h (temp @ 0 øC) by adding 98h
        skip vf = 00    ; skip on no carry
        jp tempou1      ; there was a carry, so jump
        v1 = 00         ; the temperature is below zero øC
        v1 - v0         ; get the negative temperature by
                        ; subtracting from 00
        v1 to v0        ; save into v0
```

6.7 Chip, een zelfbouw computertje

```
         p = tempmin      ; set pointer to minus sign
         ld a,a           ; put on display
tempoul  p = a-stack      ; set pointer to a-stack
         v1 = 19          ; and
         v0 * v1 to mp    ; multiply raw by 19h
         v1 = 3f          ; then
         v0 = mp/v1       ; divide by 3fh
         v0 to 3dec mp    ; convert to decimal
         p + 1            ; point to tens
         ld b,c           ; and display temperature in øC
         ret
temotxt  asciz "Temp out=  ?? ²C"
tempmin  asciz "-"
; note: change 6F (o) in hexfile into DF (LCD degree sign)
;
; daylite, subroutine to measure and display daylight value from LDR
; LDR connected to input 3
; the value is only indicative of the light
;
daylite p = daylit2       ; point to text
         ld 0,f           ; display text
         v0 = ff          ; ff = xor mask
         v1 = ana 3       ; v1 = light value
         v0 xor v1        ; take complement of light value
         p = a-stack
         v0 to 3dec mp    ; convert to decimal
         ld d,f           ; put on display
         skip out 9 = 1   ; skip if light automatic is on
         ret              ; not on, return
;
; check measured light value against light value low (v7) and
; high (v8)
; when measured < low then set light on,
; when measured > high then set light off
;
         v0 to v1         ; copy light value into v1
         v0 - v7          ; v0 = measured - low
         skip vf = 00     ; skip if v0 > v7
         jp daylit1       ; jump to set light on
         v1 - v8          ; v1 = measured - high
         skip vf = 00     ; skip if v1 > v8
         ret              ; do nothing: low > measured >= high
         res out 0        ; set light off
         ret
daylit1  set out 0        ; set light on
         ret
daylit2  asciz "Lightvalue =    "
;
; dattime, subroutine to compare the running clock against a
; date-time string
; pointed to by the pointer:
; p -> weeks, days, hours, minutes, seconds (all values hex!)
```

6.7 Chip, een zelfbouw computertje

```
; when a dattim string byte is 7f, it will not be tested
; when the condition is met for the first time, vf will be set
; as flag (<> 00)
; and bit 7 of weeks, date-time string will be set. As soon as
; the condition is no more valid this bit wil be reset.
;
dattime  v0,v4 = mp      ; move string date and time into v0...v4
         vf = 7f         ; strip bit 7 from v0
         v0 and vf
         skip v0 <> 7f   ; skip if weeks <> 7f
         jp dattim1      ; weeks = 7f, so do not test
         vf = weeks      ; get clock weeks
         skip v0 = vf    ; skip if equal
         jp dattim7      ; not equal, jump
dattim1  skip v1 <> 7f   ; skip if days <> 7f
         jp dattim2      ; days = 7f, so do not test
         vf = days       ; get clock days
         skip v1 = vf    ; skip if equel
         jp dattim7      ; not equal, jump
dattim2  skip v2 <> 7f   ; skip if hours <> 7f
         jp dattim3      ; hours = 7f, so do not test
         vf = hours      ; get clock hours
         skip v2 = vf    ; skip if equal
         jp dattim7      ; not equal, jump
dattim3  skip v3 <> 7f   ; skip if minutes <> 7f
         jp dattim4      ; minutes = 7f, so do not test
         vf = minutes    ; get clock minutes
         skip v3 = vf    ; skip if equal
         jp dattim7      ; not equal, jump
dattim4  skip v4 <> 7f   ; skip if seconds <> 7f
         jp dattim5      ; seconds = 7f, so do not test
         vf = seconds    ; get clock seconds
         skip v4 = vf    ; skip if equal
         jp dattim7      ; not equal, jump
;
; dattim string is equal to the clock
; when this happens for the first time, bit 7 of v0 (weeks) will be 0,
; it must be set to 1 and vf must be made <> 00
;
dattim5  v0, v0 = mp
         vf = 80         ; vf is bitmask
         vf and v0       ; strip d6...d0
         skip vf = 00    ; skip if d7 = 0
         jp dattim6      ; jump, this dattim skip has alread been taken
         vf = 80         ; vf is .or. bit 7
         v0 or vf        ; set 7 of v0
         v0, v0 to mp    ; write back to dattim string
         skip a          ; skip always, let vf remain 80
dattim6  vf = 00         ; entry point for skip already taken, reset
                         ; flag
         ret
dattim7  v0, v0 to mp    ; write back to dattim string
```

6.7 Chip, een zelfbouw computertje

```
        vf = 00         ; reset flag vf
        ret
;
        org 7f0
; preset values are loaded at start-up
;       v7 = 64         ; light value low= 100d
;       v8 = 82         ; light value high= 130d
;       v9 = 10         ; temp low= 16d øC
;       va = 14         ; temp med= 20d øC
;       vb = 16         ; temp high= 22d øC
;       vc = 14         ; 'set' temperature= 20d øC
;       vd = 00         ; seconds sequencer= 00
;       ve = ff         ; selector= ff (idle)
;
;               v7v8v9vavbvcvdve
initvar bytes 64821014161400ff
```

Figuur 4/6.7.8-10: Het complete Homesys.asm programma voor de automatisering van een huis.

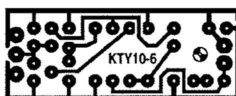
Door middel van het keyboard kunnen instellingen worden "overruled". Door op * te drukken, komt men in eerste menukeuze "Set temperature?". Door nogmaals op * te drukken verschijnt de volgende menukeuze "Start egg-timer?", gevolgd door respectievelijk "Set light on/of?", "Set alarm on/of" en "Set date & time?". Bij deze laatste optie wordt direct geschreven naar registers in de microcontroller.
Een keuze wordt bevestigd door op # te drukken. Zolang deze ingedrukt blijft is de ingestelde waarde te zien. Na loslaten kan, waar toepasselijk, een waarde worden ingevoerd. De eierwekker kan worden ingesteld op maximaal 99 minuten en 99 seconden, zolang hij loopt verschijnt de resterende tijd op het display.
Met menukeuze "Set alarm on/of?" kan het buitendeuralarm op scherp worden gezet.
Na aanzetten heeft men vijf minuten de tijd om het huis te verlaten. Daarna zal het openen van de buitendeur het alarm doen afgaan.
"Set temperature" en "Set light on/off" zijn tijdelijke instellingen, namelijk tot de volgende van toepassing zijnde tijdstring geldig wordt.

Opmerking
Een belangrijk aspect bij de verwerking van de toetsinvoer is dat de hoofdlus van het programma altijd blijft doorlopen. Dit deel van het programma is wat ondoorzichtig om direct te begrijpen, het opstellen van een stroomdiagram maakt alles veel duidelijker. De ingestelde waarden en tijden zullen voor sommige lezers niet direct van toepassing zijn, maar dat is nu juist het aardige van dit systeem; men kan alles aanpassen en het programma kan gemakkelijk worden uitgebreid. Op de "buitenwacht" maakt Chip als "Homesystem" in elk geval veel indruk.

Bob Stuurman

De listings uit dit hoofdstuk kunt u downloaden van www.vego.nl/chip

6.7 Chip, een zelfbouw computertje



Figuur 4/6.7.8-3: De print voor de temperatuursensor.

HOE MAAKT U DEZE PRINT?

OPTIE 1: zelf maken

U scant deze pagina en drukt deze met een inkjet-printer af op A4 formaat op transparante folie. U knipt de print uit en belicht er de fotogevoelige printplaat mee.

OPTIE 2: via Internet

Op **www.hobbyelektronica.nu** selecteert u uit het linker menu de optie "Printservice". In het rechter venster selecteert u het hoofdstuknummer. U kunt nu de print als TIF-file downloaden. U opent deze file in een beeldbewerkingsprogramma en drukt deze met de op de Internet-pagina aangegeven afmetingen op transparante folie af. U belicht hiermee de fotogevoelige print.

OPTIE 3: bestellen

U stuurt een **ONGEFRANKEERD** briefje naar Vego VOF, Antwoordnummer 30020, 6374 ED Landgraaf, met vermelding van het hoofdstuknummer. U krijgt per kerende post het printontwerpje op transparante folie **GRATIS** toegestuurd. U belicht hiermee de fotogevoelige print.

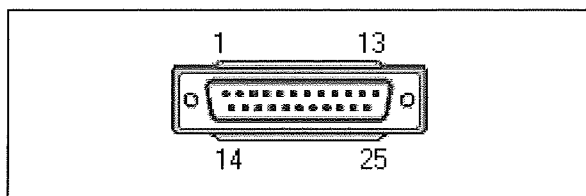
6.7 Chip, een zelfbouw computertje

4/6.8

Acht belastingen schakelen met de PC

Inleiding

Nuttig gebruik voor uw oude LPT-poort
Iedere PC heeft nog steeds een 25-polige vrouwelijke SUB-D connector, zie figuur 4/6.8-1, voorzien van een printer symbooltje. Roem uit vervlogen jaren, want moderne printers werken vrijwel allemaal met een USB-connector. Maar omdat er nog miljoenen printers in gebruik zijn, die via het Centronics protocol (LPT) communiceren, zal die SUB-D connector nog wel jaren tot de ongebruikte attributen van iedere moderne PC blijven behoren.

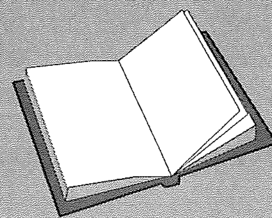


Figuur 4/6.8-1: De 25-polige vrouwelijke SUB-D connector, jaren lang dé poort waarmee uw PC communiceerde met uw printer.

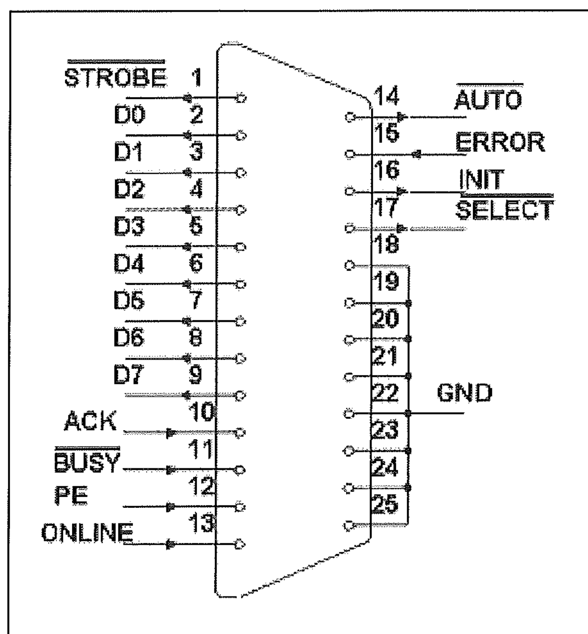
Andere toepassingen
Negen kansen op de tien gebruikt u deze Centronics connector dus niet. Jammer, want softwarematig is het heel eenvoudig om via deze connector acht datasignalen naar de buitenwereld te sturen. Die signalen kunt u natuurlijk alleen "H" en "L" aansturen, maar er zijn genoeg toepassingen waar u niets meer nodig heeft. Zelfs in een normale huiselijke omgeving zijn er nuttige automatiseringsklusjes te verzinnen, waar acht AAN/UIT-signalen heel veelbelovend klinken. In figuur 4/6.8-2 hebben wij de aansluitgegevens van die Centronics connector samengevat. U ziet de acht datasignalen D0 tot en met D7, een heleboel massa's en verder nog een aantal besturingssignalen waarmee uw PC gegevens met de printer uitwisselde. Die besturingssignalen zijn niet interessant als u alleen behoefte heeft aan de datasignalen.

Op hard- en software komt het aan
Goed, we kunnen dus acht digitale signalen aan- en uitschakelen. Maar daarvoor

LEES OOK:
Hoofdstuk 4/6.3
Hoofdstuk 4/6.4



hebben wij natuurlijk software nodig en wat hardware, die de acht D-signalen uit uw PC haalt.

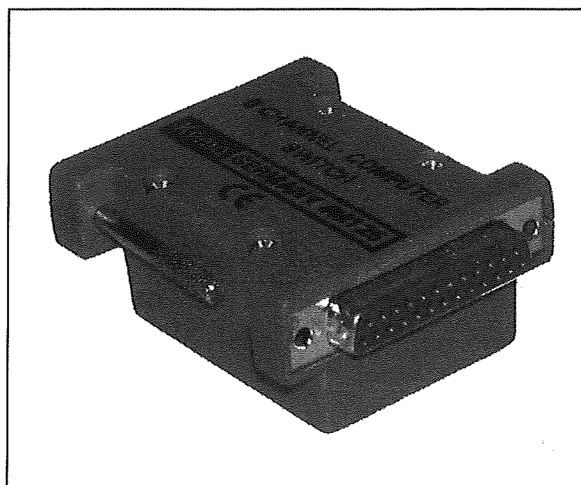


Figuur 4/6.8-2: De aansluitgegevens van de 25-polige Centronics connector.

Door de Duitse firma Kemo Electronic wordt al meer dan tien jaar een module aangeboden, die u op de printerpoort kunt aansluiten en die acht halfgeleiderrelais bevat. Die relais worden bestuurd door de D-signalen en u kunt op de uitgangen belastingen aansluiten.
Het probleem is dat er bij deze module een floppy met uitsluitend oeroude DOS-software wordt geleverd, waar u in het moderne Windows-tijdperk weinig mee kunt. Maar sinds kort biedt Kemo op haar internetsite supermoderne Windows-software aan, waarmee u de module zeer uitgebreid kunt besturen.
Dank zij deze nieuwe software wordt dit oeroude stukje elektronica opeens weer heel attractief voor de moderne elektronicus.

De Kemo module M125

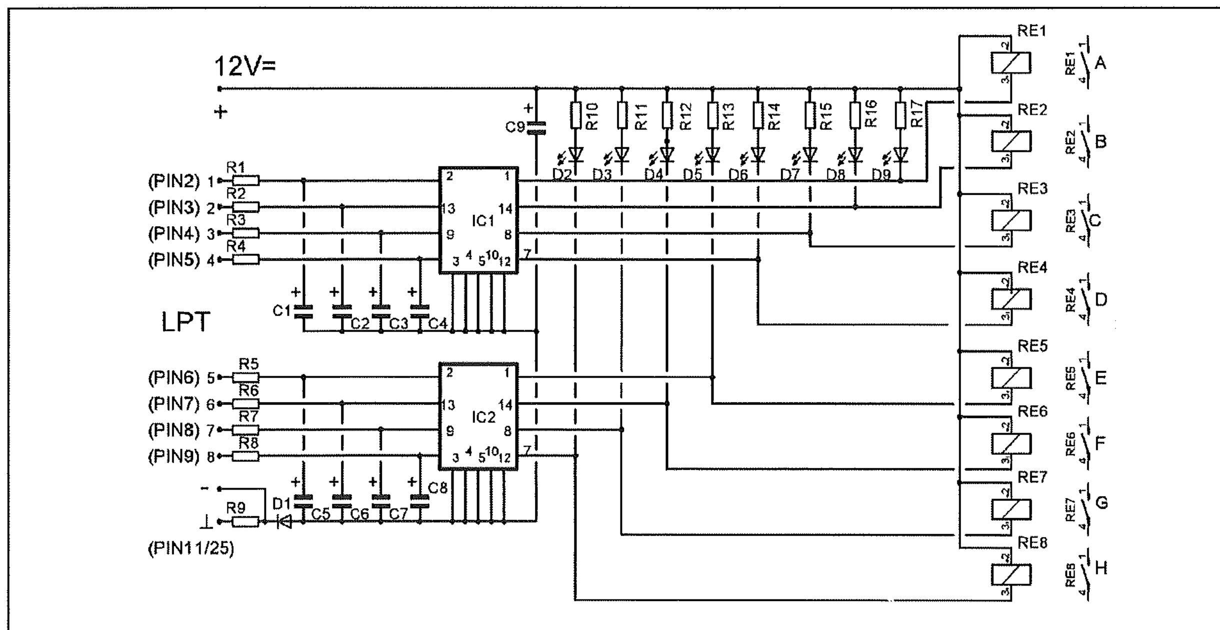
Inleiding
De Kemo module M125, zie figuur 4/6.8-3, is een klein kastje met aan weerszijden 25-polige SUB-D connectoren. De mannelijke kunt u in de Centronics connector van uw PC pluggen, op de vrouwelijke staan de relaisuitgangen ter beschikking. De module wordt aangeboden voor € 33,18, misschien niet écht goedkoop, maar het alternatief van zelfbouw in zo'n klein kastje is nauwelijks aanwezig.



Figuur 4/6.8-3: De Kemo module M125.

Specificaties
De specificaties van de M125 zijn als volgt:
- aantal uitgangen: acht solid state relais
- schakelspanning relais: 6 V min., 40 V max.
- soort spanning: gelijk- of wisselspanning
- schakelstroom relais: 400 mA DC, 200 mA AC
- bediening: Windows-compatibele software vanaf versie 95

6.8 Acht belastingen schakelen met de PC



Figuur 4/6.8-4: Het schema van de vrijwel identieke B210 zelfbouw set.

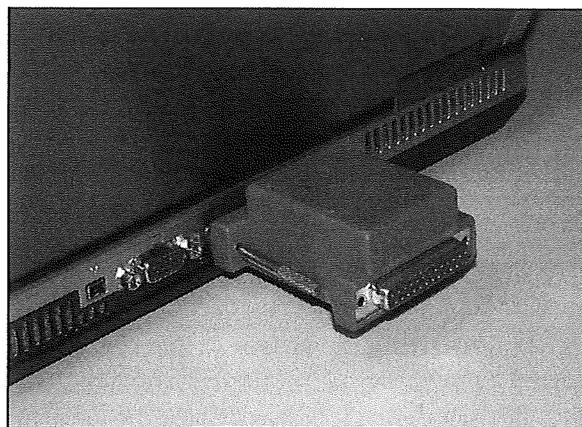
- afmetingen:
  73 mm x 56 mm x 29 mm
- gewicht:
  82 g

De interne elektronica

Als u de module openschroeft zult u vaststellen dat de elektronica is ingegoten en dus niet toegankelijk is. Maar Kemo levert, behalve kant-en-klare modules, ook bouwpakketten. De B210 is een vrijwel identieke schakeling en het ligt voor de hand dat de elektronica in de module in grote lijnen gelijk is aan deze van het bouwpakket. U kunt dus iets verwachten als voorgesteld in figuur 4/6.8-4.
De acht datalijnen gaan naar twee IC'tjes, ongetwijfeld buffers. De uitgangen van de buffers besturen de ingangen van de solid-state relais.

Aansluiten op uw PC

Dank zij de werkelijk zeer kleine afmetingen van de module zult u het kastje zonder problemen op uw PC kunnen aansluiten. Het kastje past zelfs, zie figuur 4/6.8-5, zonder problemen op een laptop. Met de twee schroefstangen kunt u de module verankeren aan het chassis van de PC.

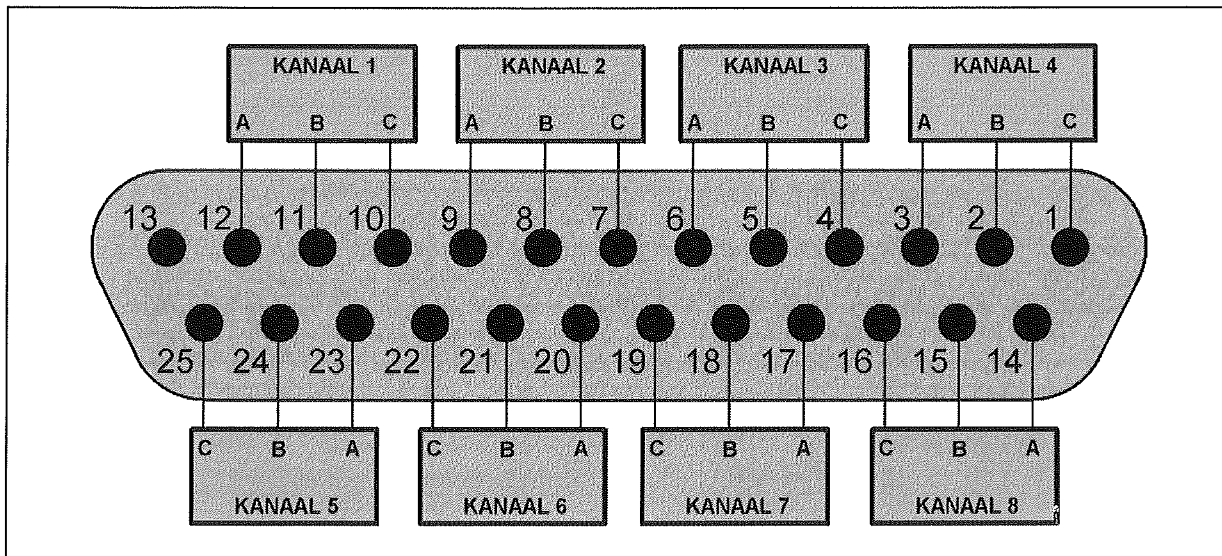


Figuur 4/6.8-5: De Kemo module M125 is aangesloten op een laptop.

Het aansluiten van de belastingen

De module heeft géén voeding, de solid state relais worden op de een of andere manier uit een van de besturingssigna-

6.8 Acht belastingen schakelen met de PC



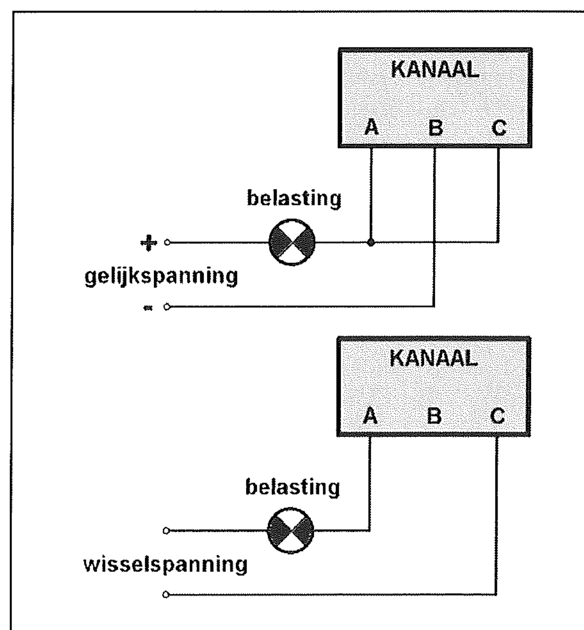
Figuur 4/6.8-6: De aansluitingen van de acht uitgangskanalen op de module.

len op de Centronics connector gevoed. U moet dus extern voeden. De acht kanalen hebben ieder drie aansluitingen A, B en C die volgens figuur 4/6.8-6 op de vrouwelijke SUB-D connector ter beschikking staan.
Zoals uit de technische specificaties blijkt, kunt u met de module zowel gelijk- als wisselspanning schakelen en wel tussen 6 V en 40 V. Het schakelschema van één kanaal is voorgesteld in figuur 4/6.8-7. Voedt u met gelijkspanning, dan worden de aansluitingen A en C parallel geschakeld en gaan dan naar de belasting. B is de retourleiding naar de massa. Voedt u met wisselspanning, dan wordt aansluiting B niet gebruikt en bouwt u een simpele seriekring op tussen de voedingsspanning, de belasting en de A/C van de module.

Netspanning schakelen

Wilt u netspanningsbelastingen schakelen, dan moet u als belasting een relais opnemen, waarvan de contacten in staat zijn de 230 V netspanning te schakelen.

Als u heel gevoelige relais toepast, die maar weinig stroom verbruiken, dan kunt u in serie met de relaisspoel een LED opnemen, zodat u een optische indicatie krijgt van in- en uitgeschakelde uitgangen.



Figuur 4/6.8-7: Het aansluiten van belasting en voeding op de module.

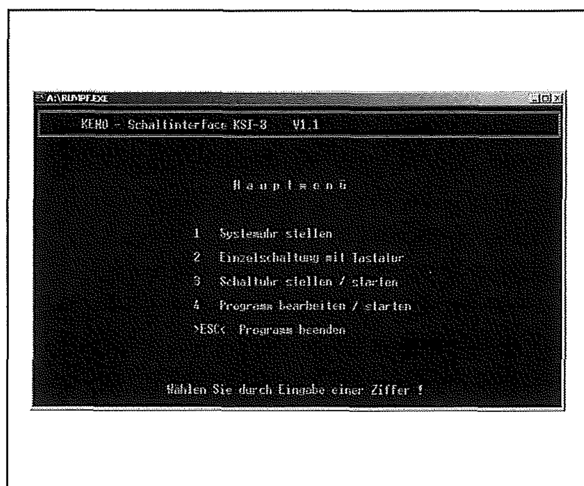
6.8 Acht belastingen schakelen met de PC

De DOS-software KSI-8

Inleiding

De module wordt geleverd met een diskette waarop een oeroud (1995!) MS-DOS programma "Schaltinterface KSI-8" te vinden is. Dit programma zit verborgen in het bestand "RUMPF .EXE", dat slechts 46 kB groot is. Die goede, oude MS-DOS tijd! Er is in principe niets in te brengen tegen een goed MS-DOS programma. Na het starten van het genoemd bestand verschijnt het openingsvenster van figuur 4/6.8-8 in een Windows-venster. Met de toetsen 1, 2, 3 en 4 kunt u een van de opties openen:

- systeemtijd instellen;
- bediening met toetsenbord;
- schakelklok instellen;
- programma starten.



Figuur 4/6.8-8: Het bij de module geleverde programma "Schaltinterface KSI-8".

Maar het via de internetsite www.kemo-electronics.com te downloaden Windows-programma biedt dezelfde functionaliteit maar uiteraard in een grafisch veel aantrekkelijker jasje. Vandaar dat we ons hierop concentreren.

De WIN-software KSI-8

Inleiding

De Windows-versie van KSI-8 kunt u downloaden van de genoemde site, maar u vindt het waarschijnlijk sneller terug op de site van "Hobby Elektronica & Actueel IC-handboek". Ga naar www.hobbyelektronica.nu en klik in het linker frame de optie "Softwareservice" aan. In het rechter frame ziet u nu meteen de link naar aanvulling 117 en hoofdstuk 4/6.8.

Het bestand "ksi8_winall.exe" is 613 kB groot en u kopieert het naar een eigen directory op uw harde schijf. De naam van het bestand, winall, doet reeds vermoeden dat u dit stukje besturingssoftware onder iedere versie van Windows kunt gebruiken.

Dubbelklikken op het bestand start de installatie, die volledig voldoet aan de Windows-standaarden. U kunt dus een bestemmingsdirectory opgegeven, een naam van de startmenu folder en al dan niet een desktop icoon laten opnemen. "ksi8_winall.exe" installeert twaalf bestanden met een totale omvang van 612 kB in de door u geselecteerde directory.

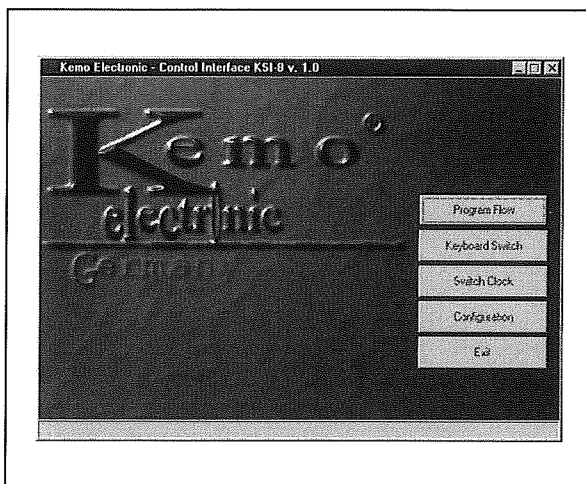
Control Interface KSI-8

Met de software "Control Interface KSI-8", zie figuur 4/6.8-9, kunt u de module M125 op diverse manieren besturen.

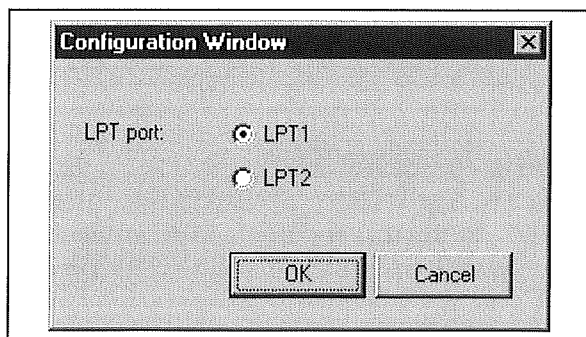
- Program Flow
  Met deze utility stelt u een aantal programmastappen in waar u de acht kanalen aan of uit stuurt. De tijdsduur van iedere stap kunt u instellen in uren, minuten, seconden en tienden van een seconde. De stappen worden achter elkaar doorlopen.

6.8 Acht belastingen schakelen met de PC

- Keyboard Switch
  Met deze utility bestuurt u de acht kanalen met de toetsen 1 tot en met 8 van uw toetsenbord.
- Switch Clock
  Met deze utility stelt u maximaal 255 programma's in, waarin u aan ieder kanaal een inschakel- en uitschakeltijd koppelt. Deze tijden zijn in te stellen in maanden, dagen, uren, minuten en seconden.



Figuur 4/6.8-9: Het openingsscherm van "Control Interface KSI-8".



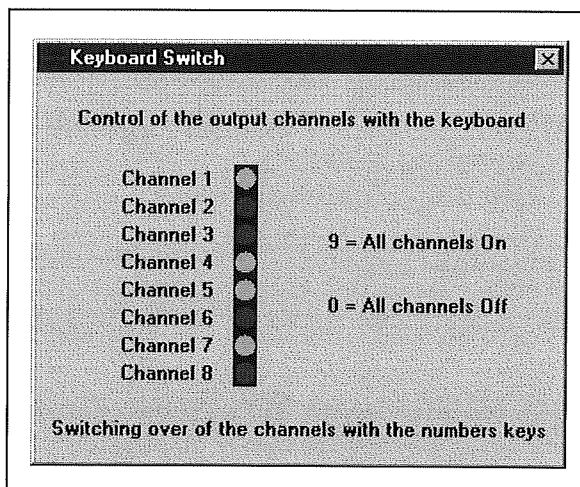
Figuur 4/6.8-10: Het instellen van de printerpoort waarop de M125 is aangesloten.

Configuration

Met deze optie stelt u, zie figuur 4/6.8-10, de LPT-poort in waarop de module is aangesloten. In vrijwel alle gevallen zal dat LPT1 zijn.

Keyboard Switch

Deze optie is, zie figuur 4/6.8-11, ideaal voor het snel testen van alle uitgangen en de schakelingen die er op zijn aangesloten. Met de toetsen 1 tot en met 8 van uw toetsenbord kunt u de uitgangen individueel in- en uitschakelen. Met toets 9 schakelt u alle kanalen aan, met toets 0 uit. Een nuttige utility!

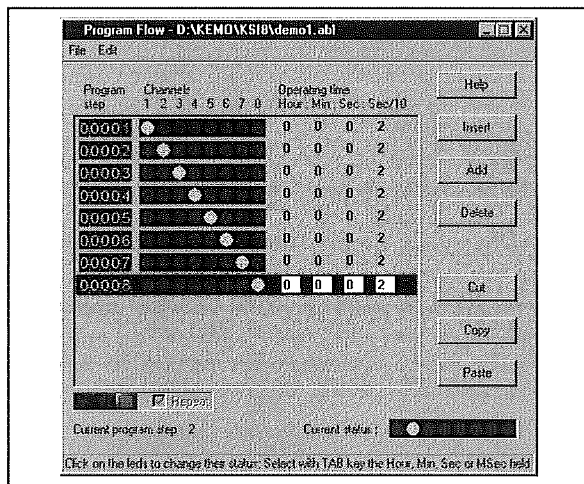


Figuur 4/6.8-11: Het met het toetsenbord bedienen van de acht schakelkanalen.

Program Flow

Met deze optie, zie figuur 4/6.8-12, kunt u een programma samenstellen van in totaal 99.999 stappen. In iedere stap kunt u alle uitgangen individueel in- of uitschakelen. Daarnaast kunt u de tijdsduur van iedere stap instellen in uren, minuten, seconden en tienden van een seconde. In het gegeven voorbeeld hebben wij een eenvoudig looplichtje geprogrammeerd. In iedere stap wordt maar één uitgang AAN gestuurd en alle stappen van het programma duren 0,2 seconde.

6.8 Acht belastingen schakelen met de PC



Figuur 4/6.8-12: Het instelvenster van de optie "Program Flow".

Met de rechter knoppen kunt u snel met het programma werken:

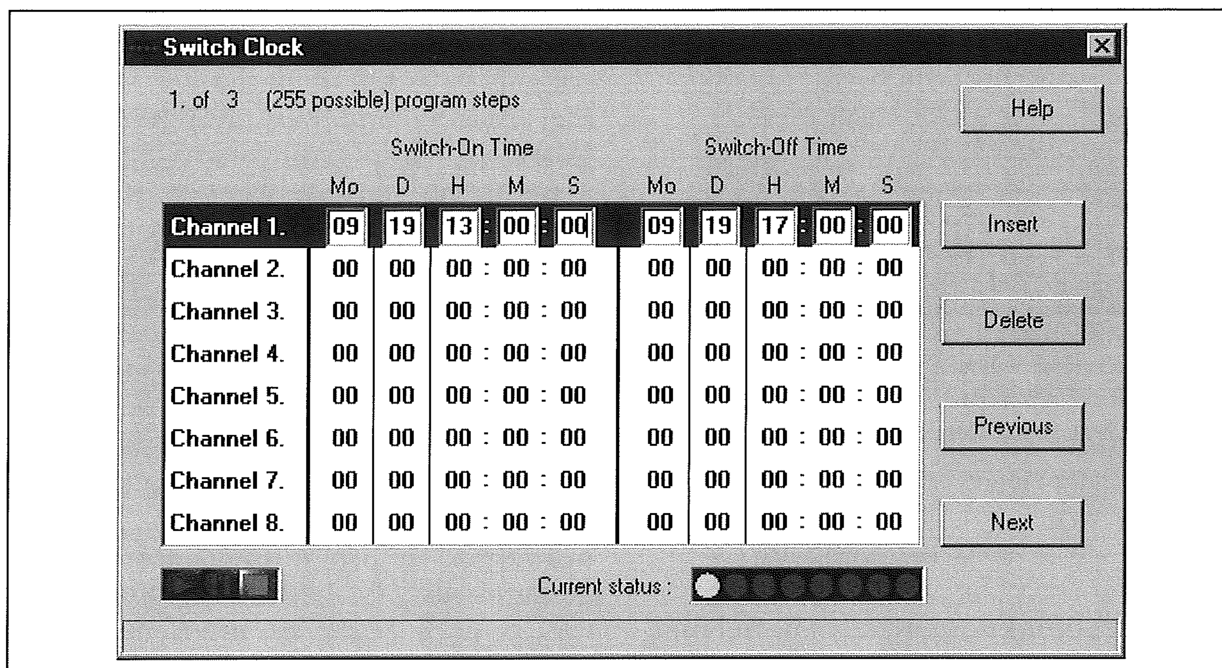
- Add:
  Voegt een stap aan het programma toe, deze wordt opgenomen na de laatste stap.
- Insert:
  Voegt een stap in het programma in vóór de stap die met de muiscursor is aangeklikt.
- Delete:
  Verwijdert de met de muiscursor aangeklikte stap.
- Cut:
  Idem, maar de stap wordt opgeslagen in een buffer.
- Copy:
  Slaat de met de muiscursor geselecteerde stap in een buffer op.
- Paste:
  Voegt de in de buffer opgeslagen programmastap toe aan het programma.

U selecteert een stap door met de linker muisknop te klikken op het tellertje, links in de kolom. Nadien klikt u met de linker muisknop op de "LED'jes" om de kanalen aan of uit te schakelen. In de vier witte vakjes, die in de blauwe achtergrond worden uitgespaard, kunt u vervolgens de tijden invoeren.
Met de drie onderste knoppen "Play", "Halt" en "Stop" kunt u vervolgens het programma starten, stoppen of afbreken. Door het aanklikken van de knop "Repeat" wordt het programma in een eeuwigdurende lus afgespeeld. In de LED-balk, rechts onder in het venster, ziet u de status van de acht uitgangen in de actuele stap van het programma.
Met het "File"-menu kunt u een programma opslaan op uw harde schijf of reeds bewaarde programma's weer oproepen. De programma's worden opgeslagen als .ABL-bestanden in de directory waar u "Control Interface KSI-8" heeft geïnstalleerd.

Switch Clock

Deze laatste optie van "Control Interface KSI-8" heeft indrukwekkende mogelijkheden. De "Program Flow" is een sequentieel programma. Alle door u geprogrammeerde stappen worden één na één afgewerkt. De "Switch Clock" is een serieel programma dat wordt bestuurd door de kalenderchip die in uw PC zit. U kunt in totaal 255 programmastappen programmeren. In iedere programmastap, zie figuur 4/6.8-13, kunt u aan ieder van de kanalen een bepaalde starttijd en stoptijd toekennen. Maar die tijden worden nu in de gaten gehouden door de kalender.
In stap 1 kunt u bijvoorbeeld programmeren dat kanaal 1 op dinsdag, 5 oktober om 15 uur 20 minuten en 2 seconden moet inschakelen en op vrijdag, 29 oktober om 10 uur, 15 minuten en 10 seconden weer moet uitschakelen. Op dezelfde manier kunt u de zeven overige kanalen programmeren. Maar in de volgende stap van het programma kunt u

6.8 Acht belastingen schakelen met de PC



Figuur 4/6.8-13: Met "Switch Clock" krijgt u een indrukwekkende timer waarmee u de acht kanalen door de kalender-chip van uw PC kunt laten besturen.

volledig andere start- en stopgegevens invoeren.
Het invoeren van de gegevens gaat op de volgende manier:

- Vakje Mo:
  Hier voert u de maand in van 01 (januari) tot 12 (december).
- Vakje D:
  Hier voert u de dag in van 1 voor zondag tot 7 voor zaterdag. Door in dit vakje een X toe te voegen, bijvoorbeeld 2X, weet het programma dat het uw bedoeling is dat deze stap wekelijks op de gedefinieerde dag wordt herhaald.
- Vakje H:
  Hier voert u het uur in van 00 tot 23. Voert u hier het getal 24 in, dan zal de programmastap ieder uur worden herhaald.
- Vakjes M en S:
  Hier voert u uiteraard de minuten en de seconden in van 00 tot en met 59.

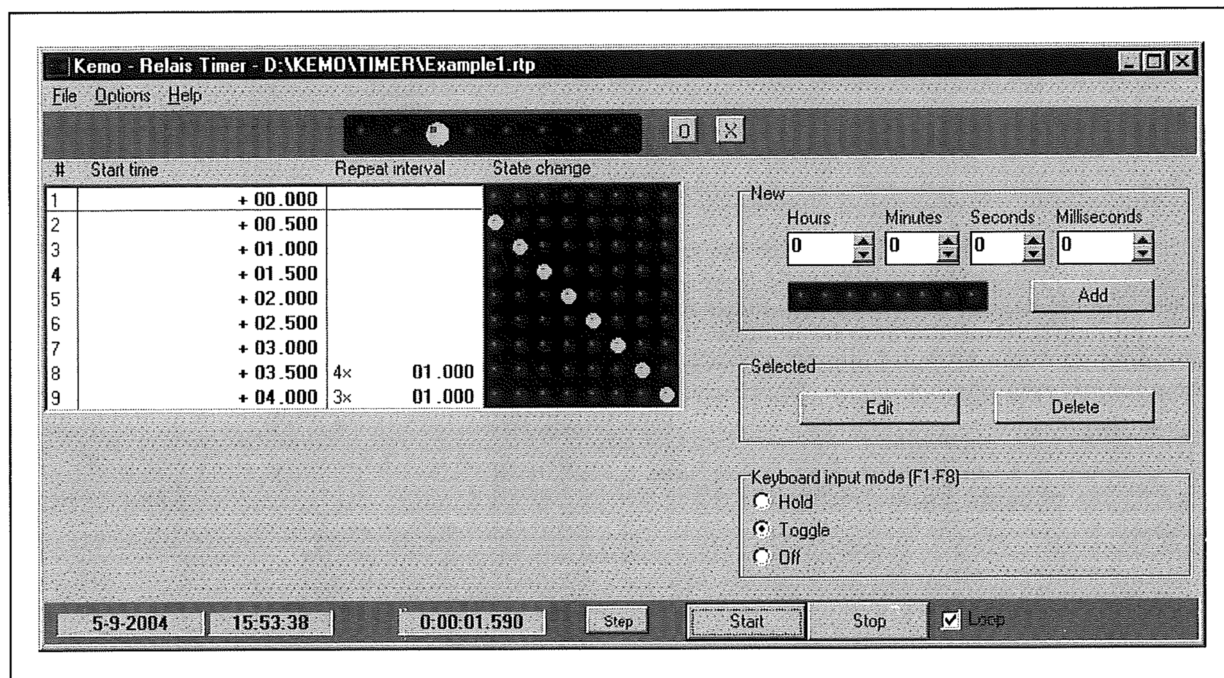
Met de drie onderste knoppen kunt u het programma activeren, bevriezen of stoppen. Er bestaat geen mogelijkheid diverse programma's onder een eigen naam op te slaan. Als u "Switch Clock" afsluit kunt u het programma weliswaar bewaren, maar waar en onder welke naam dat gebeurt is onduidelijk.

Kemo Relais Timer

Inleiding

Met de software "Kemo Relais Timer" kunt u de acht kanalen op een sensationele manier programmeren. Het programma werkt met programmastappen waarin u ieder van de acht uitgangen kunt inschakelen, uitschakelen of gelijk aan de situatie in de vorige stap kunt instellen. Voor ieder van de stappen kunt u een startdatum en -tijd instellen. De tijd wordt ingesteld in uren, minuten, secon-

6.8 Acht belastingen schakelen met de PC



Figuur 4/6.8-14: Het werkvenster van "Kemo Relais Timer".

den en milliseconden. Bovendien kunt u iedere stap een aantal keren herhalen (iterations) en de herhalingstijd instellen.

Downloaden

Het bestand "relaistimer11.exe" van 1.190 kB kunt u downloaden van de site van Kemo, maar u vindt ook dit bestand sneller terug op de site van "Hobby Elektronica & Actueel IC-handboek". Ga naar www.hobbyelektronica.nu en klik in het linker frame de optie "Software-service" aan. In het rechter frame ziet u nu meteen de link naar aanvulling 117 en hoofdstuk 4/6.8.

Installatie

Ook dit programma wordt op de standaard Windows-manier geïnstalleerd in een door u opgegeven directory. Er worden tien bestanden aangemaakt met een gezamenlijke omvang van 1,08 MB. Daarnaast treft u in de subdirectory "Source" de volledige sourcecode van het programma aan.

"Program Flow" plus "Switch Clock"

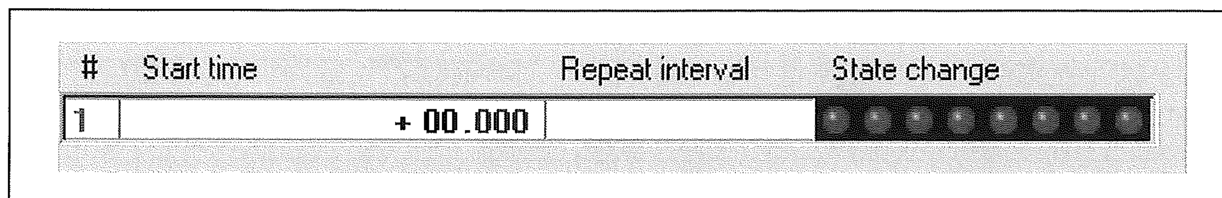
In principe is deze "Kemo Relais Timer" een combinatie van de twee reeds besproken programma's "Program Flow" en "Switch Clock". U kunt in één venster, zie figuur 4/6.8-14, zowel sequentiële als seriële stappen programmeren.

LED Indicator

Boven in het venster ziet u een grote LED-balk met acht LED's. Deze geven de momentele status van de acht uitgangen weer.

U kunt bovendien de LED's afzonderlijk in- en uitschakelen met de functietoetsen F1 tot en met F8. Met de drie opties in het kadertje "keyboard input mode (F1-F8)" kunt u de werking van dit geheel controleren.

6.8 Acht belastingen schakelen met de PC



Figuur 4/6.8-15: De gegevens van de eerste stap van het programma kunnen worden ingevoerd.

- Hold:
  De uitgangen worden AAN gestuurd zolang u een van de functietoetsen ingedrukt houdt.
- Toggle:
  De eerste keer drukken zet de uitgang AAN, de tweede keer drukken UIT.
- OFF:
  De werking van de functietoetsen wordt uitgeschakeld.

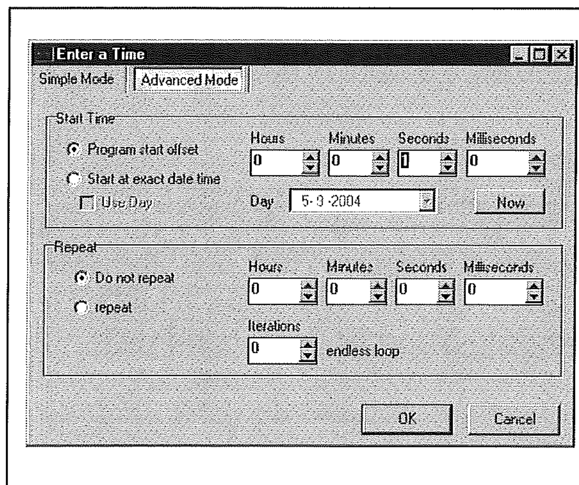
U kunt het systeem ook bedienen door met de linker muisknop op de LED's te klikken. Deze actie heeft altijd voorrang en werkt ook als u "OFF" heeft geselecteerd. Door het klikken op de knoppen "O" en "X" kunt u alle uitgangen respectievelijk UIT of AAN schakelen.

Sequentieel programmeren van de eerste stap uit het programma

Open het "File"-menu en klik de optie "New project" aan. Er verschijnt nu één lege stap in het venster, stap die wij in figuur 4/6.8-15 even vergroot voor u hebben weergegeven.

Dubbelklik ergens in deze eerste stap. Onmiddellijk verschijnt het venster "Enter a Time" van figuur 4/6.8-16 op uw scherm. Hierin kunt u de stap volledig programmeren.

Wilt u een sequentiële besturing, dan klikt u de optie "Program start offset" aan en voert in de vier vakjes daarnaast een tijd in. Let op dat dit een offset-tijd is. Dat betekent dat deze stap maar eerst na de door u ingestelde tijd na het starten van het programma actief wordt! Vult u hier bijvoorbeeld een seconde in, dan zal deze stap één seconde na het starten van het programma worden geactiveerd. Vervolgens moet u programmeren hoe de acht uitgangen zich moeten gedragen in deze stap. Rechts in figuur 4/6.8-15 ziet u acht LED's "State change" die u door er op te klikken met de linker muisknop kunt in- of uitschakelen. Maar daarnaast kunt u ook een klein pijltje instellen. Dat wil zeggen dat de uitgang bij het activeren van deze stap niet van status zal veranderen, dus gelijk zal blijven aan de status van de vorige stap.

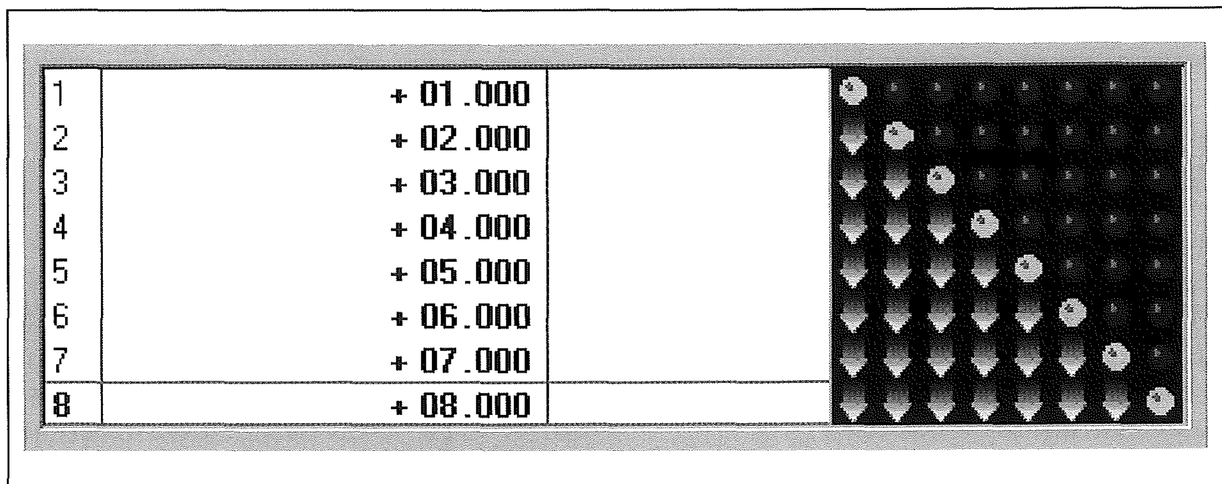


Figuur 4/6.8-16: In dit venster programmeert u iedere stap van het programma.

Een stap toevoegen

Klik op de knop "Add" in het kader "New" om de tweede stap van uw programma toe te voegen.

6.8 Acht belastingen schakelen met de PC



Figuur 4/6.8-17: Een eenvoudig voorbeeld van sequentiële programmering.

TIJD	UIT 1	UIT 2	UIT 3	UIT 4	UIT 5	UIT 6	UIT 7	UIT 8
START	UIT	UIT	UIT	UIT	UIT	UIT	UIT	UIT
START + 1 s	AAN	UIT	UIT	UIT	UIT	UIT	UIT	UIT
START + 2 s	AAN	AAN	UIT	UIT	UIT	UIT	UIT	UIT
START + 3 s	AAN	AAN	AAN	UIT	UIT	UIT	UIT	UIT
START + 4 s	AAN	AAN	AAN	AAN	UIT	UIT	UIT	UIT
START + 5 s	AAN	AAN	AAN	AAN	AAN	UIT	UIT	UIT
START + 6 s	AAN	AAN	AAN	AAN	AAN	AAN	UIT	UIT
START + 7 s	AAN	AAN	AAN	AAN	AAN	AAN	AAN	UIT
START + 8 s	AAN	AAN	AAN	AAN	AAN	AAN	AAN	AAN

Figuur 4/6.8-18: Het resultaat van het programma van figuur 4/6.8-17.

Op dezelfde manier kunt u in het venster van figuur 4/6.8-16 deze stap programmeren. Let op de offset-programmering! Iedere ingevoerde tijd geldt vanaf de start van het programma!

Een eenvoudig voorbeeld

In figuur 4/6.8-17 hebben wij een eenvoudig voorbeeld van een sequentiële besturing geprogrammeerd. Alle uitgangen worden een na een aangestuurd met een onderlinge tijdsoffset van een seconde. Omdat wij pijltjes-symbolen hebben ingevoerd, zullen de uitgangen na de eerste inschakeling steeds AAN blijven. Het resultaat van deze programmering is samengevat in de tabel van figuur 4/6.8-18.

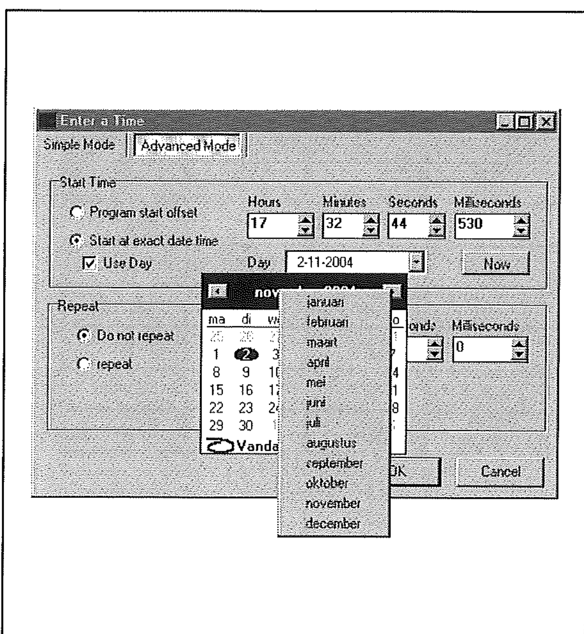
Loops programmeren

Een nuttige optie in "Kemo Relais Timer" is de mogelijkheid om aan iedere stap een "Repeat"-functie toe te kennen. In het venster van figuur 4/6.8-16 schakelt u de optie "Do not repeat" uit en klikt de knop "Repeat" aan. U kunt nu een herhalingstijd en het aantal herhalingen instellen. Let op dat de herha-

lingstijd ook als offset-tijd geldt, dus na start van het programma!

Kalender programmeren

Aan iedere stap kunt u een actuele datum koppelen. Ga weer naar het venster van figuur 4/6.8-16 en klik "Start at exact date time" aan.
U kunt nu de actuele datum en tijd kiezen met de knop "Now" of via het bekende Windows-kalendertje een datum invullen, zie figuur 4/6.8-19.



Figuur 4/6.8-19: Het invullen van een exacte starttijd voor de stap uit het programma.

Programma uitvoeren

Met de knoppen "Start", "Stop" en "Loop" kunt u het programma eenmalig starten, er een eindeloze lus van maken of het programma onderbreken.

Instellingen opslaan

Via het menu "File" kunt u uw programmering opslaan onder een eigen naam en weer in het programma inladen.

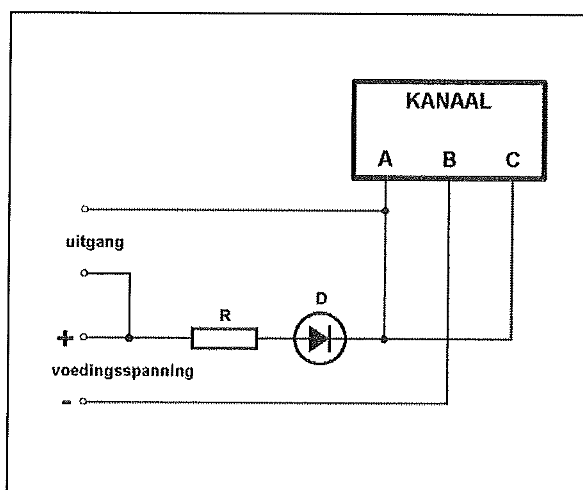
Een handig printje

Inleiding

De kleine afmetingen van de module hebben natuurlijk een bepaalde charme, maar zijn niet zo handig als u iets op de uitgangen moet aansluiten. De 25 pennetjes van de SUB-D connector zitten wel érg dicht bij elkaar. Vandaar dat wij een handig hulpprintje hebben ontworpen dat u in de uitgangsconnector van de M125 kunt pluggen en waardoor u gemakkelijk toegang krijgt tot de acht uitgangen.

Het schema

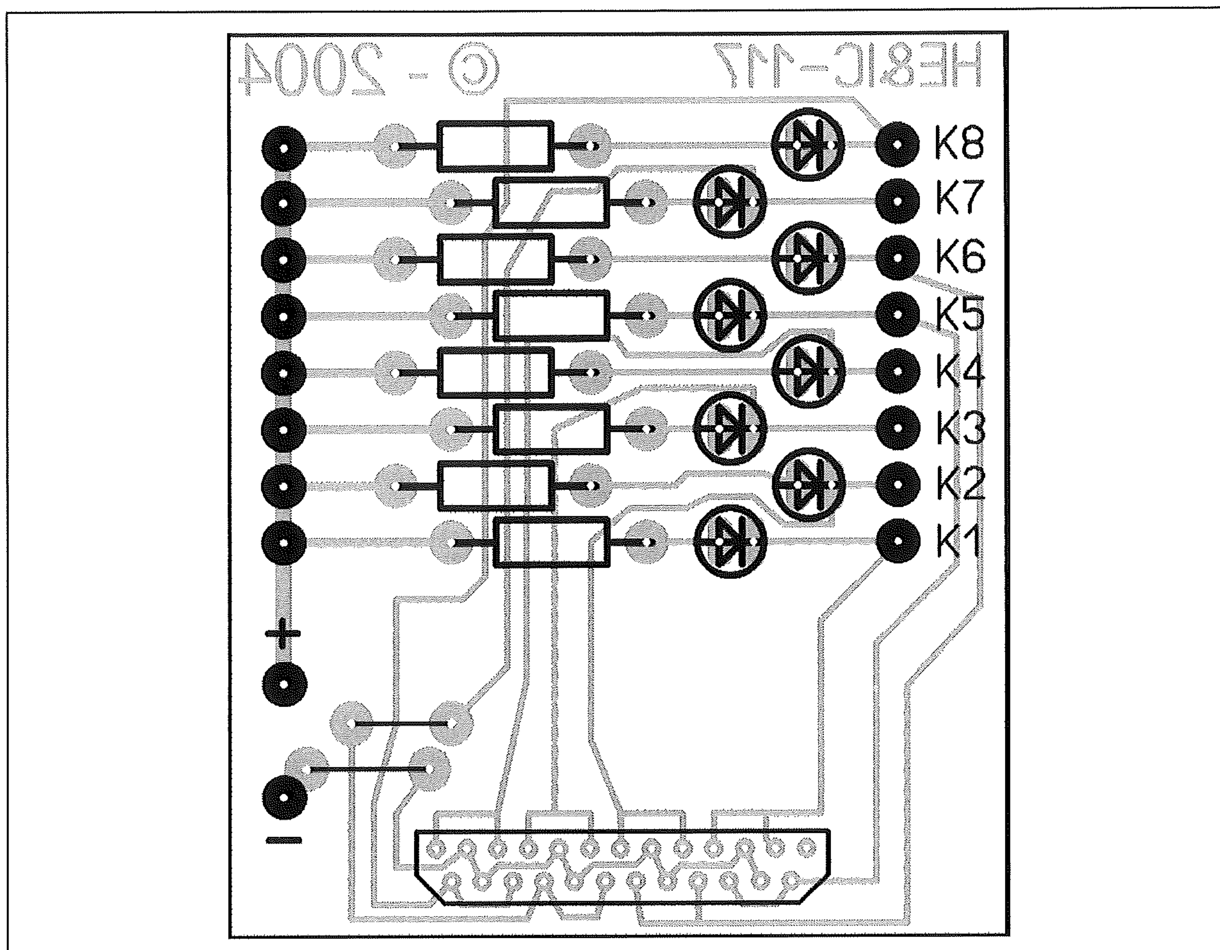
Het schema, getekend in figuur 4/6.8-20, is de eenvoud zelve. Tussen de plus van de voeding en de uitgangen A en C van een kanaal staat de serieschakeling van een weerstand en een LED. De waarde van de weerstand is afhankelijk van de voedingsspanning.



Figuur 4/6.8-20: Het schema voor één kanaal op de hulpprint.

Voor 12 V kunt u bijvoorbeeld 1 kΩ toepassen, er vloeit dan een stroom van ongeveer 10 mA door de LED. U krijgt een optische indicatie van de modus van de

6.8 Acht belastingen schakelen met de PC



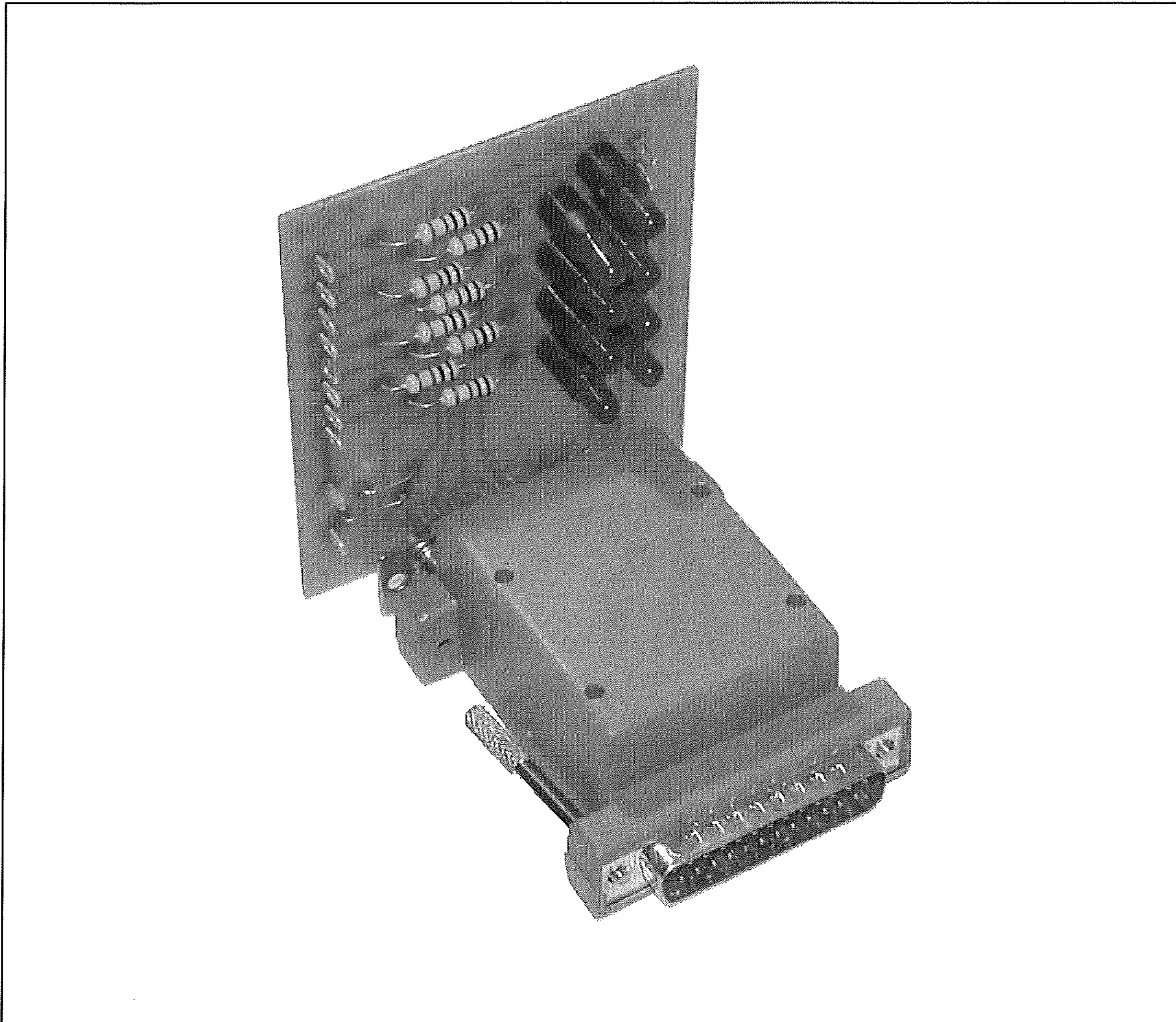
Figuur 4/6.8-22: De componentenopstelling van de hulpprint.

acht uitgangen. Over deze serieschakeling wordt de belasting van het kanaal aangesloten. Deze staat dus tussen de plus van de voeding en de uitgang van een kanaal. Via punt B wordt de uitgang naar de massa getrokken. Als voeding kunt u een netstekkervoeding toepassen die een spanning van 12 V levert.

De print

De print is voorgesteld in figuur 4/6.8-21 op de laatste pagina van dit hoofdstuk. De componentenopstelling volgt uit figuur 4/6.8-22. Met enig geluk vindt u een mannelijke 25-polige SUB-D connector voor printmontage. Deze kunt u rechtstreeks in de print solderen. Als u een type met soldeeroogjes vindt moet u op alle 25 oogjes een draadje solderen en deze nadien een na een in de print wurmen. Vastsolderen en het lijkt alsof u een printmodel hebt gesoldeerd! Wij hebben de LED's op 5 mm lange kunststof afstandsbusjes gezet, waardoor het printje er keurig komt uit te zien.
Op de foto van figuur 4/6.8-23 ziet u hoe u de Kemo module op de print kunt pluggen. De acht uitgangen kunt u nu aansluiten tussen de soldeerlipjes van de voeding (links) en de kanaaluitgangen (rechts).

6.8 Acht belastingen schakelen met de PC



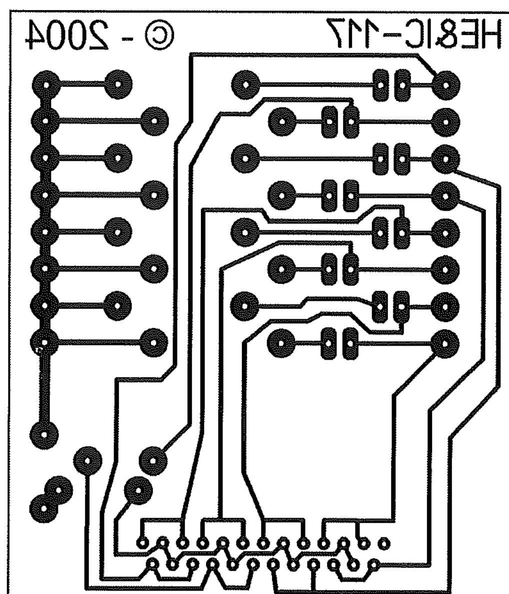
Figuur 4/6.8-23: De Kemo module en de hulpprint vormen een gemakkelijk toegankelijk geheel.

Nadere gegevens

De Kemo module M125 is via Internet te bestellen en is uit voorraad leverbaar door:

Vego VOF, Postbus 32014, 6370 JA Landgraaf (NL)
Telefoon: 045-533.22.00
Fax: 045-533.22.02
E-mail: vego_vof@compuserve.com
Internet: www.vego.nl/kemo

6.8 Acht belastingen schakelen met de PC



Figuur 4/6.8-21: De print voor de hulpschakeling.

HOE MAAKT U DEZE PRINT?

OPTIE 1: zelf maken

U scant deze pagina en drukt deze met een inkjet-printer af op A4 formaat op transparante folie. U knipt de print uit en belicht er de fotogevoelige printplaat mee.

OPTIE 2: via Internet

Op **www.hobbyelektronica.nu** selecteert u uit het linker menu de optie "Printservice". In het rechter venster selecteert u het hoofdstuknummer. U kunt nu de print als TIF-file downloaden. U opent deze file in een beeldbewerkingsprogramma en drukt deze met de op de Internet-pagina aangegeven afmetingen op transparante folie af. U belicht hiermee de fotogevoelige print.

OPTIE 3: bestellen

U stuurt een **ONGEFRANKEERD** briefje naar Vego VOF, Antwoordnummer 30020, 6374 ED Landgraaf, met vermelding van het hoofdstuknummer. U krijgt per kerende post het printontwerpje op transparante folie **GRATIS** toegestuurd. U belicht hiermee de fotogevoelige print.

6.8 Acht belastingen schakelen met de PC